UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Arquitectura Híbrida para el estándar de compresión de vídeo H.265/HEVC

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

David Guillermo Fernández Herrera

Directores

Guillermo Botella Juan
Alberto Antonio del Barrio García

Madrid

UNIVERSIDAD COMPLUTENSE DE MADRID

Arquitectura Híbrida para el estándar de compresión de vídeo H.265/HEVC

Tesis Doctoral
David Guillermo Fernández Herrera

Dirigida por los Doctores:
Guillermo Botella Juan
Alberto Antonio del Barrio García

Madrid, 2019

Arquitectura Híbrida para el estándar de compresión de vídeo H.265/HEVC

Memoria que presenta para optar al título de Doctor en Informática:
David Guillermo Fernández Herrera

Dirigida por los doctores:
Guillermo Botella Juan
Alberto Antonio del Barrio García

Universidad Complutense de Madrid
Facultad de Informática

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D. David Guillermo Fernández Herrera, estudiante en el Programa de Doctorado D9BK-Doctorado en Ingeniería, de la Facultad de Informática de la Universidad Complutense de Madrid, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada Arquitectura Híbrida para el estándar de compresión de vídeo H.265/HEVC y dirigida por los doctores Guillermo Botella Juan y Alberto Antonio del Barrio García.

DECLARO QUE:

La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita.

Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada de conformidad con el ordenamiento jurídico vigente.

En Madrid, a 1 de Julio de 2019.

A mi familia, y al resto de hadas y duendes que en mi vida me acompañan.

Agradecimientos

Agradezco a José Valladares y a Javier Lizuain permitirme realizar las investigaciones en Prodys y de esta forma, poder compaginar los estudios y mi vida laboral.

A Gaspar Gómez, gracias por ser mi mentor y por tu colaboración en las investigaciones.

A Guillermo Botella, sin ti ni siquiera hubiera empezado este trabajo y mucho menos lo habría terminado. Gracias por tu gran ayuda y tu apoyo.

A Alberto Antonio del Barrio, gracias por tu ayuda incondicional y tus sabios consejos.

A Daniel Santamaría, Daniel Orellana, Conchi Rodríguez, Santiago Gómez y al resto de mis compañeros de Prodys, porque hacéis que ir a trabajar no suponga esfuerzo alguno.

A Begoña Rivas, miña raíña.

A mi padre, aprendiz de filósofo, gracias por ser un modelo a seguir. Gracias también porque nadie ha mostrado más interés que tú en que finalice con éxito este trabajo.

A mi madre, gracias por tu amor y paciencia. Te quiero.

A mis hijos Daniel y David, y a Laura, sin vosotros no conocería el amor verdadero.

David Guillermo Fernández Herrera
En Madrid, Julio de 2019

Abstract

In the last few decades, multimedia applications have made a considerable amount of progress. Video coding standards have noticeably contributed to this advance by becoming essential during data transmission. Since 1980, video standards have focused on reducing the size of the resulting bitstream. Besides this objective, the maintenance of a high level of visual quality has been the second factor to consider when trying to optimize video transmission. Due to the increasing demand for digital content, nowadays it is vital to optimize the available bandwidth for transmission and reception of video data, targeting high compression data rates without a significant quality loss. Around 20 years ago video compression was something of a novelty, today it is ubiquitous. From TVs to cell phones and camcorders to PCs, most consumer electronic devices rely on video compression. While most users assume that these improvements are a result of general-purpose advances in consumer technology, they are actually all driven by video compression.

Since the late 1980s, the International Telecommunication Union (ITU) has developed several coding standards for real-time transmission applications. High Efficiency Video Coding (HEVC), aka H.265, is the next step forward in video compression, achieving better compression efficiency than previous standards. This new coding standardization codec is a project of the Joint Collaborative Team on Video Coding (JCT-VC), a collaboration between the ITU-T Video Coding Expert Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG) organizations. The main goal of HEVC is to retain the same quality even while consuming less than 50% of the bitstream size when compared to the previous H.264 standard. The bitstream size reduction means that, at the same quality, a compressed video sequence should occupy less storage space or employ less bandwidth in a transmission. This allows making better use of the video distribution channels. Expressed in an equivalent way, for the same storage size, the quality of a compressed video should be better than that corresponding to using previous standards. This has, however, been achieved by considerably increasing the complexity of the algorithm and, consequently, high power consumption and a high economic cost to meet the real time constraints of the high definition (HD) and ultra-high definition (UHD) television resolutions.

Video compression standards define the syntax or format of a compressed video sequence and a decoding method, but the design of the encoder is not standardized. Decoders do not provide quality or compression efficiency, as their main function is to obtain a video from a standard bitstream. Compression efficiency, as well as video quality, directly depends on the encoder implementation, making its design a critical task. For this reason, it is important to understand that not all encoders are equally implemented and the compression efficiency can vary a lot in each encoder. Thus, in this PhD Thesis, we will focus on the design of an efficient HEVC encoder flow.

In this work, temporal and spatial homogeneity-based reductions in the computational cost of HEVC encoding are proposed. Both are implemented by means of spatial and temporal homogeneity analysis and classification, which are directly applied to the input image so they may be implemented independently or just integrated inside the encoding loop. This image pre-analysis is performed using logic units and embedded hardware on a GPU, so the computational cost associated with this process is zero for the processor in charge of the encoding process. After locating the spatially and temporally homogeneous

regions, several improvements in the encoding process are performed in order to allow fast encoding with a negligible bitstream increase and while maintaining the quality.

The state-of-the-art methods perform the quality assessment by just using PSNR. However, the encoding should leverage human perception when trying to optimize the encoding process in order to achieve a better visual experience from the user's perspective. In this PhD Thesis, the quality assessment of the all proposed optimizations and the determination of several thresholds have been performed using metrics based on the human visual system (HVS).

According to HVS, textured areas can be coded with a higher amount of noise, before the noise becomes noticeable. Furthermore, HVS is more sensitive to details in areas with homogeneous texture activities, so it is annoying to observe artefacts in these areas. In this work, the proposal consists in improving the quality of the regions with homogeneous textures to obtain a better global perception from the HVS. After this classification is conducted, a pre-processing filter is applied to regions with a chaotic distribution to reduce the amount of detail that the encoder needs to process. The pre-processed frames are then handed to the encoder. Finally, a visual quality improvement is obtained after the HEVC encoding thanks to the proposed algorithms.

Most of works related to HEVC encoding have been implemented on the HEVC test model (HM). This software is typically employed for evaluating new techniques within the HEVC standard, but it is a non-optimized software. Despite of this, HM must be used to conduct a fair comparison with other works, since it allows to establish a set of common test conditions. Because of the promising results achieved after the integration of our work in this software and after comparing our proposal with several references, we decided to create a complete video compression system based on implementing the whole flow within an HEVC encoder capable of supporting real time execution. Heterogeneous computing opens new ways to tackle computational intensive problems using low-power and low-cost platforms. Thus, we decided to use a FPGA-GPU-microprocessor environment to test our approach in real time conditions, but the suggested ideas could easily be deployed in other encoders supporting real time regardless of the target hardware. The FPGA is used to extract video and audio from a Serial Digital Interface (SDI) input, then the valid data is transfer to the GPU through a PCIe bus. The GPU is used for pre-processing purposes and to perform the temporal and spatial homogeneity analysis and classification on the image. The microprocessor is the device in charge of the encoding process. The information retrieved from the image is shared between GPU and microprocessor. Lastly, the optimized HEVC encoder flow is executed on the microprocessor side.

All in all, to the best of our knowledge there is no flow considering spatially and temporally homogeneous regions and realizing such analyses on the input frames to offload the processor by means of the GPU. Furthermore, a high performance FPGA-GPU-microprocessor platform for real-time execution is proposed. Finally, HVS-based metrics are utilized to verify the performance of our flow.

Keywords:

HEVC, H.265, spatial homogeneity, temporal homogeneity, HVS, GPU, FPGA, real-time encoder, texture analysis.

Resumen

En las últimas décadas las aplicaciones multimedia han evolucionado enormemente. Los estándares de compresión de vídeo han contribuido notablemente a este avance convirtiéndose en esenciales para la transmisión de datos multimedia. Desde 1980, los estándares de compresión de vídeo han enfocado sus principales esfuerzos a reducir la tasa de datos generada y proporcionar un alto nivel de calidad visual. Debido a la creciente demanda de contenido digital, hoy en día resulta de vital importancia optimizar el ancho de banda utilizado para la transmisión de vídeo digital, buscando obtener grandes tasas de compresión de datos sin pérdidas apreciables en la calidad del servicio. Veinte años atrás se consideraba a la compresión de vídeo como algo novedoso, actualmente es algo ubicuo. La mayoría de dispositivos que constituyen la electrónica de consumo hacen uso de una manera u otra de la compresión de vídeo. Aunque la mayoría de usuarios asume que estas mejoras son el resultado de los avances generales que se han producido en la electrónica de consumo, lo cierto es que son principalmente debidas a la compresión de vídeo.

Desde finales de la década de los ochenta, la organización *International Telecommunication Union (ITU)* ha desarrollado varios estándares de codificación de vídeo para aplicaciones de transmisión en tiempo real. *High Efficiency Video Coding (HEVC),* también conocido como H.265, supone un gran salto tecnológico en la compresión de vídeo, proporcionando mejor eficiencia en la compresión respecto a estándares predecesores. Este nuevo códec es un proyecto del grupo *Joint Collaborative Team on Video Coding (JCT-VC),* una colaboración entre las organizaciones *ITU-T Video Coding Experts Group (VCEG)* e *ISO/IEC Moving Picture Experts Group (MPEG)*. El principal objetivo que persigue HEVC respecto a su predecesor H.264, consiste en reducir la tasa de datos generada en un 50% para un nivel de calidad preestablecido. Una reducción en la tasa de datos generada para una misma calidad visual implica que una secuencia de vídeo codificada, ocupa menos espacio de almacenamiento y requiere para su transmisión de un ancho de banda menor, lo que permite optimizar el uso de los canales de distribución de vídeo. Expresado de una manera equivalente, para la misma tasa de datos generada, la calidad de la secuencia de vídeo codificada debe superar a la obtenida empleando estándares de codificación previos. Sin embargo, esta mejora en las prestaciones ha sido obtenida a expensas de incrementar considerablemente la complejidad del algoritmo y consecuentemente, requerir mayor consumo de potencia y mayor coste económico para cumplir los requerimientos que permiten la codificación en tiempo real de señales de televisión de alta y ultra-alta definición.

Los estándares de compresión de vídeo definen la sintaxis o formato que debe seguir una secuencia de bits que define un vídeo y el método a seguir para realizar la descodificación, pero el diseño del codificador está fuera de su ámbito de aplicación. Los descodificadores no proporcionan ni calidad visual ni eficiencia en la compresión, puesto que su principal función es obtener un vídeo para su visualización a partir de un conjunto de bits que cumple con la sintaxis de un determinado estándar. Tanto la tasa de compresión como la calidad visual del vídeo, dependen directamente de la implementación realizada en el codificador, haciendo de su diseño una tarea crítica. Por esta razón, es importante entender que no todos los codificadores están implementados de la misma manera, pudiendo variar la eficiencia en la codificación enormemente entre dos implementaciones

distintas de un codificador bajo un mismo estándar. Debido a todo lo expuesto, la presente Tesis Doctoral está orientada al diseño de un flujo eficiente de codificación HEVC.

En el trabajo realizado, se proponen una serie de optimizaciones sobre el flujo de codificación HEVC basadas en la detección de regiones espacial y temporalmente homogéneas. El análisis y la posterior clasificación de este tipo de regiones son aplicados sobre la imagen de entrada al codificador, por lo que pueden ser implementados como un proceso independiente a la codificación o como procesos integrados en el bucle de codificación. Como el pre-análisis de la imagen es aplicado haciendo uso de unidades lógicas y de módulos de hardware dedicado de una GPU trabajando como co-procesador, el coste computacional asociado a este proceso es prácticamente cero para el procesador encargado de realizar la codificación propiamente dicha. Después de localizar las regiones espacial y temporalmente homogéneas, sobre las zonas clasificadas en alguna de estas categorías, se aplican una serie de algoritmos de optimización que permiten acelerar la codificación a expensas de un incremento inapreciable en la tasa de datos generada y proporcionando un alto nivel de calidad visual.

En el estado del arte, la evaluación de la calidad visual se realiza empleando únicamente la métrica PSNR. Sin embargo, en la optimización del proceso de codificación se debe de tener en cuenta cómo funciona la percepción humana para que el usuario final del sistema viva una mejor experiencia visual. En la presente Tesis Doctoral, los procesos de evaluación de las prestaciones de todas las optimizaciones propuestas y los asociados a la determinación de los umbrales utilizados, han sido llevados a cabo empleando métricas con consideraciones perceptuales basadas en el sistema visual humano (*Human Visual System, HVS*).

Conforme al funcionamiento del HVS, en las áreas que presentan texturas complejas puede introducirse mayor cantidad de distorsión debida a la codificación que en otro tipo de regiones de la imagen, antes de que dicha distorsión sea perceptible. Por otro lado, HVS es más sensible a percibir los detalles de la imagen en áreas con texturas homogéneas, por lo que resulta muy molesto observar artefactos en este tipo de regiones. En la propuesta realizada, se mejora la calidad de las zonas de la imagen que presentan texturas homogéneas para obtener una mejor calidad visual global de la secuencia de vídeo desde la perspectiva del HVS. Una vez que la clasificación se ha realizado, se aplica un filtrado a las regiones que presentan una distribución caótica para reducir la cantidad de detalle que el codificador debe procesar, lo que reduce el número de bits generado. Las imágenes pre-procesadas son enviadas al codificador, donde gracias a los algoritmos propuestos, se consigue una mejora de calidad global de la secuencia de vídeo tras aplicar la codificación.

La mayoría de trabajos relacionados con la codificación HEVC son implementados en el modelo de test de HEVC conocido como HM. Este software es empleado para la evaluación de nuevas técnicas dentro del estándar HEVC, pero se trata de un software completamente sin optimizar, haciendo imposible alcanzar ejecución en tiempo real. A pesar de ello, HM debe de ser utilizado para realizar una comparativa justa con otros trabajos existentes, puesto que permite establecer un conjunto de condiciones de test comunes. Debido a que se obtuvieron resultados muy prometedores tras la integración del trabajo realizado en este software, y después de comparar la propuesta realizada con diversas referencias, se decidió crear un sistema completo de compresión de vídeo en tiempo real. Para cumplir este objetivo se implementó el flujo completo dentro de un codificador HEVC capaz de soportar ejecución en tiempo real. El cómputo heterogéneo permite explorar nuevos caminos para abordar problemas computacionalmente muy intensivos utilizando plataformas de bajo consumo y bajo coste. Por ello, se decidió utilizar una plataforma basada en FPGA-GPU-microprocesador para verificar el funcionamiento de las propuestas realizadas bajo condiciones de ejecución en tiempo real. Aunque se ha

escogido esta plataforma, las propuestas realizadas pueden ser fácilmente implementadas en cualquier dispositivo hardware. La FPGA es utilizada para extraer tanto el vídeo como el audio de una entrada *SDI (Serial Digital Interface).* Tras realizar la extracción, los datos válidos son transferidos a la GPU por medio de un bus PCIe. La GPU es utilizada para llevar a cabo una etapa de pre-procesado de la imagen y para realizar el análisis y posterior clasificación de las regiones espacial y temporalmente homogéneas existentes en la señal de vídeo de entrada. El microprocesador es el dispositivo encargado de llevar a cabo el proceso de codificación HEVC. La información extraída por la GPU relativa a las características presentes en las imágenes de entrada, es compartida con el microprocesador. Finalmente, conforme a esta información, el flujo de codificación HEVC optimizado es ejecutado en el microprocesador.

En definitiva, hasta donde sabemos, no existe un flujo de codificación que considere optimizaciones basadas en la detección de regiones espacial y temporalmente homogéneas, analizando únicamente las imágenes de entrada a la codificación y cuya implementación haya sido realizada utilizando módulos de hardware dedicado y unidades lógicas de una GPU. Además, se han verificado las prestaciones que ofrece el flujo de codificación propuesto en un entorno de ejecución en tiempo real basado en FPGA, GPU y microprocesador, y se han utilizado para la caracterización de los algoritmos métricas perceptuales conforme al funcionamiento del HVS.

Palabras clave:

HEVC, H.265, homogeneidad espacial, homogeneidad temporal, HVS, GPU, FPGA, codificación en tiempo real, análisis de texturas.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Motivación y justificación

Debido al gran auge de la distribución de contenidos digitales por la creciente demanda existente por parte de la sociedad actual, ha surgido la necesidad de optimizar el ancho de banda disponible para la transmisión/recepción de datos, buscando grandes tasas de compresión de datos sin pérdidas asociadas a la calidad del servicio. La visualización de contenidos de alta resolución a través de Internet, videoconferencias, la televisión digital tanto terrestre como la distribuida a través de satélite, la emisión de noticias en directo por medio de tecnología móvil 3G/4G, videovigilancia, distribución de contenidos multimedia bajo demanda, etc. son sólo un breve listado de las tecnologías existentes actualmente en las que se hace uso de la compresión de vídeo digital.

El principal objetivo perseguido por la compresión de vídeo es proporcionar la mayor calidad visual posible haciendo uso de una mínima cantidad datos. Para tal propósito se han empleado durante décadas los estándares de codificación de vídeo MPEG-2, MPEG-4 y H.264 (AVC o MPEG-4 Part10) suponiendo grandes avances, en cuanto a reducción de la tasa de datos se refiere, cada nuevo estándar respecto a sus antecesores. En el año 2013 se finalizó la primera versión del estándar *High Efficiency Vídeo Coding (HEVC)*, aprobado por ISO/IEC bajo el nombre de norma ISO/IEC 23008-2 (MPEG-H part 2). El principal atractivo de HEVC es la reducción de tasa de datos de hasta un 50% respecto a H.264 para una misma calidad visual prefijada. Desde otro punto de vista, debido a esta reducción de datos que proporciona HEVC respecto a H.264, para una misma cantidad de datos prefijada HEVC es capaz de proporcionar mayor calidad visual y por lo tanto, mejorar la experiencia vivida por el usuario que al final de la cadena visualiza el contenido digital. Otra ventaja

que ofrece la reducción de la tasa de datos utilizada, es la disminución de los costes asociados al almacenamiento de contenidos digitales o los asociados a la transmisión de contenidos a través de enlaces por satélite o tecnología móvil 3G/4G/5G, donde se factura en función del número de datos utilizado. HEVC posibilita la emisión de canales en resolución 4K, lo cual hasta su aparición era considerado como totalmente inviable por la cantidad de datos necesaria. Teléfonos móviles, tabletas, ordenadores, equipos profesionales de distribución y contribución de contenidos para televisión, videocámaras y gran cantidad de dispositivos se benefician junto a las tecnologías anteriormente nombradas de las mejoras de HEVC respecto a sus predecesores. Los estándares de compresión de vídeo digital permiten la interoperabilidad entre los dispositivos creados por distintos fabricantes, definiendo de manera inequívoca la sintaxis que debe de seguir el conjunto de bits generados tras aplicar la compresión y el proceso de descodificación que permite obtener a partir de esos bits nuevamente un vídeo digital.

El proceso que recibe como entrada una señal de vídeo digital y obtiene como salida un conjunto de bits que describen dicho vídeo, se denomina codificación. La codificación de vídeo se centra en la explotación de redundancias existentes en el vídeo sobre el que se aplicará la compresión. La redundancia puede ser espacial, temporal o en el dominio de la frecuencia. La redundancia espacial explota la correlación existente entre los píxeles de una misma imagen mientras que la redundancia temporal, se centra en la explotación de las similitudes existentes entre las distintas imágenes que componen un vídeo. Por otro lado, el sistema visual humano presenta ciertas limitaciones, siendo menos sensible a percibir artefactos debidos a la codificación en imágenes que presentan gran contenido de alta frecuencia, por lo que conociendo sus limitaciones y la correlación existente entre las distintas componentes de frecuencia puede incrementarse la tasa de compresión. Por lo tanto, es en el proceso de codificación donde se obtiene la reducción de la tasa de datos generada, pero como se ha comentado, los estándares de compresión de vídeo definen el proceso de descodificación y no el de codificación. El estándar HEVC ofrece la disponibilidad de un gran número de técnicas y herramientas para realizar la compresión, pero es en la parte codificadora donde se decide cuáles emplear o cómo hacer uso de ellas. Centrándose en la estimación de movimiento para ilustrar este concepto, el estándar define como un vector que describe el movimiento de una parte de la imagen debe de ser convertido a un conjunto de bits, pero no define cómo dicho vector ha sido obtenido por el algoritmo de estimación de movimiento. Cuanto mejor sea la definición del movimiento existente, mayor calidad visual se proporcionará y menor número de bits será generado, por lo que el diseño de una búsqueda de movimiento eficiente y eficaz resulta de vital importancia en el proceso de codificación. Al igual que ocurre en el proceso de búsqueda de movimiento que explota la redundancia temporal, sucede en los procesos de codificación encargados de explotar la redundancia espacial y la del dominio de la frecuencia.

La mejora de prestaciones conseguida por HEVC en la codificación se consigue por medio de un considerable aumento en la carga computacional. Por este motivo, desde la reciente aparición del estándar HEVC, la reducción de la complejidad del proceso de codificación intentando no afectar a sus prestaciones, ha sido objeto de numerosos estudios y publicaciones.

1.2 Objetivos

El principal objetivo que se persigue en la presente Tesis Doctoral, es mejorar la eficiencia en el proceso de codificación HEVC, posibilitando que la compresión de vídeo pueda ser ejecutada en tiempo real. Dicha eficiencia puede ser definida como la capacidad de reducir la carga computacional asociada a la codificación, sin afectar a la facultad del codificador de minimizar la tasa de datos generada respecto a estándares predecesores y sin pérdida asociada a la calidad visual de la imagen descodificada.

Como objetivos específicos que se derivan del objetivo principal y que se han cumplido en las diferentes fases de la investigación se enumeran los siguientes:

- Estudio exhaustivo de la norma ISO/IEC 23008-2 (MPEG-H part 2) para entender todas las herramientas y técnicas disponibles en el estándar HEVC.
- Evaluación del estado del arte y establecimiento de los ámbitos en los que se pueden realizar las aportaciones más relevantes.
- Diseño, evaluación e implementación de algoritmos que permitan reducir la complejidad de diferentes técnicas empleadas en la codificación HEVC con tal grado de abstracción que puedan ser implementados en diversas plataformas hardware.
- Facilitar el diseño de dispositivos que lleven incorporado un codificador HEVC, centrándose en los requerimientos de Prodys S.L, empresa con la que se participa en un proyecto de investigación y desarrollo para renovar sus equipos de contribución de vídeo basados en el estándar H.264.
- Establecimiento de la plataforma hardware más adecuada para la implementación de los algoritmos.
- Obtención y evaluación de los resultados, así como comparación con los trabajos más relevantes de la literatura existente.

Como consecuencia del cumplimiento de los objetivos fijados se han diseñado e implementado una serie de algoritmos que ha permitido implementar un codificador HEVC en una plataforma heterogénea de bajo coste y consumo. El sistema desarrollado proporciona gran calidad visual y una importante reducción de la tasa de datos generada.

1.3 Metodología y medios utilizados

En la etapa inicial de la investigación se estableció un marco teórico, por medio de la consulta de la literatura relacionada con la compresión de vídeo utilizando el estándar HEVC, analizando y describiendo el contexto asociado a la línea de trabajo escogida. Dentro de la memoria se indican las referencias consultadas y las causas por las que se ha decidido emplearlas o descartarlas. Por otro lado, resulta fundamental para la consecución con éxito de los objetivos planteados, realizar un profundo estudio sobre la norma ISO/IEC 23008-2 para conocer las técnicas que se utilizan para llevar a cabo la compresión de vídeo digital, así como otros documentos que resuman de una manera más amena y general los principios básicos de funcionamiento de esta normativa. Existe una implementación práctica de la norma ISO/IEC 23008-2, que incluye tanto un codificador como un

descodificador conocida como software de referencia HM. Se trata de una implementación en lenguaje C++ sin optimizar, cuyo principal objetivo es clarificar los conceptos teóricos que aparecen en la normativa. Este software es empleado como referencia en la mayoría de investigaciones a la hora de evaluar nuevos algoritmos o implementaciones, ya que aporta un caso base en cuanto a tasa de datos generada, calidad de la imagen y tiempos de ejecución se refiere. Este software está lejos de permitir una ejecución en tiempo real, pero proporciona una cota sobre la mínima tasa de datos que se puede obtener para una determinada calidad visual. Para localizar las técnicas computacionalmente más costosas dentro del estándar HEVC se realizaron medidas de tiempos de ejecución sobre el software HM. Antes de la creación del banco de pruebas que permitió la obtención de los datos pertinentes, se inició una labor de recopilación de una serie de secuencias de vídeo de diferentes resoluciones que sirven como señal de vídeo a codificar. Las secuencias presentan diferentes características de manera que puedan representar una muestra significativa de las distintas condiciones que se pueden presentar como entrada a la codificación, es decir, secuencias con mucho movimiento, secuencias estáticas, busto parlante, tráfico rodado, naturaleza, texturas homogéneas o con gran complejidad, etc. Las secuencias están disponibles para descarga libre lo que permite que futuros estudios sobre el trabajo realizado puedan verificar los resultados obtenidos. Se han empleado secuencias que han sido utilizadas en la literatura relacionada con la investigación y que posibilitan la comparación con trabajos previos. Tras la obtención de dichas secuencias se establecieron unas condiciones de test para realizar una batería de pruebas exhaustiva aplicando distintas configuraciones. Para evaluar la calidad visual desde distintas perspectivas se han estudiado las diferentes métricas que existen y que son utilizadas en trabajos similares. Existen dos grandes grupos de métricas de calidad: las denominadas objetivas y las subjetivas. Las primeras abarcan medidas tradicionales como *PSNR (Peak Signal-to-Noise Ration)* o *SSIM (Structural Similarity Index)* y aquellas medidas que pretenden modelar el funcionamiento del sistema visual humano. La utilización de las métricas objetivas permite obtener unos datos repetitivos, pero los resultados obtenidos distan de representar con certeza la opinión de un ser humano, problema tratado dentro de la literatura existente y hecho que se ha verificado empíricamente. La medida de calidad visual de un vídeo es intrínsecamente subjetiva, debido a que el sistema visual humano así lo es y todavía no se ha conseguido obtener una métrica que modele su funcionamiento con exactitud. En el segundo grupo de métricas, se incluyen aquellas métricas que se basan en puntuaciones u opiniones dadas por usuarios o expertos tras visualizar los vídeos descodificados, en definitiva, se trata de evaluar la experiencia vivida por el usuario final del sistema de compresión. Para establecer unos criterios comunes y unas condiciones generales de visualización existen normas al respecto, siendo una de las más utilizadas la norma ITU-R BT.500.

Una vez clasificadas las distintas técnicas por su complejidad, se realizó una consulta de la literatura existente sobre la optimización de dichas técnicas, centrándonos en los ámbitos donde poder realizar una aportación significativa. Una vez estudiado el estado del arte, se procedió al diseño de un primer conjunto de algoritmos con tal grado de abstracción que se permitiese su implementación en diversas plataformas hardware. Se estudiaron las virtudes y carencias de cada arquitectura hardware para finalmente realizar la implementación en una plataforma en concreto. Tras verificar el correcto funcionamiento de los algoritmos, se procedió a la obtención de medidas iniciales de calidad visual, de tasa de datos generada y tiempos de ejecución empleando el banco de pruebas preestablecido, de manera que quedasen completamente caracterizados los algoritmos propuestos. Tras evaluar los resultados y comprobar el cumplimiento de los objetivos preestablecidos, se obtuvieron diversas conclusiones, que condujeron a la realización de ajustes sobre el

diseño, a un replanteamiento del problema y a la implementación de modificaciones sobre los algoritmos hasta alcanzar las metas planteadas. Para la aceleración de las pruebas realizadas empleando el software de referencia se utilizaron las máquinas multi-núcleo disponibles en el *Group of Architecture and Technology of Computing Systems (ArTeCS)* de la Universidad Complutense de Madrid. Para verificar que las optimizaciones realizadas se aplican correctamente en la codificación se requiere el uso de analizadores de tramas HEVC. Estos analizadores permiten comprobar las diferentes elecciones llevadas a cabo en la codificación.

Tras verificar que los algoritmos diseñados ofrecen ventajas sobre trabajos previos realizados sobre la misma temática, se decidió implementar los algoritmos en un sistema de codificación en tiempo real al existir una importante carencia al respecto en la literatura existente. Para implementar dicho sistema se requiere el uso de un codificador optimizado que soporte la ejecución en tiempo real. Se optó por una usar una implementación existente de software libre de manera que se pudiesen reproducir los resultados obtenidos fácilmente, y en un software propietario de una empresa colaboradora que aporta unas prestaciones muy superiores al anterior, al estar orientado a un uso profesional en el mercado de contribuciones de vídeo digital.

1.4 Aportaciones

HEVC está basada en una arquitectura híbrida basada en la división de la imagen en los bloques que suponen la unidad básica de codificación. Dicha arquitectura combina una predicción basada en la explotación de las redundancias temporales y espaciales, con una codificación basada en transformadas y cuantificación escalar para eliminar redundancias en el dominio de la frecuencia, y una posterior codificación entrópica binaria. La predicción que explota la redundancia espacial es conocida como predicción intra, mientras que la que se basa en la redundancia temporal se conoce como predicción inter. El resultado de la predicción se resta a la imagen de entrada, obteniendo el error de predicción, que a su vez, sirve de entrada al proceso de transformación. La salida del proceso de transformación se conoce como residuo, siendo este residuo cuantificado antes de introducirse en la codificación entrópica.

Si bien esta arquitectura ya fue utilizada en estándares de compresión de vídeo previos como H.264, en HEVC se mejoran todos los procesos involucrados en la codificación. HEVC introduce un nuevo proceso de división de la imagen recursivo que permite adaptarse con mayor precisión a la forma de los objetos presentes en la imagen a codificar, lo cual permite reducir la tasa de datos generada considerablemente, a expensas de un gran aumento en la carga computacional. En el caso de la predicción intra, el descodificador obtiene una predicción del bloque codificado a partir de píxeles espacialmente cercanos previamente descodificados. El proceso a seguir para obtener dicha predicción se indica mediante el denominado modo de predicción intra, que debe de ser indicado por el codificador tras evaluar el conjunto de modos de predicción que permite el estándar utilizado. En HEVC los modos de predicción intra son 35, mientras que en H.264 se permitía únicamente un máximo de hasta 9 modos. En lo referente a la predicción inter, HEVC incluye numerosas mejoras respecto a H.264, entre ellas destaca la inclusión de nuevos tamaños de bloque para definir el movimiento existente en el vídeo a codificar y de nuevos algoritmos para determinar el movimiento de un bloque a partir de vecinos tanto

espaciales como temporales. La obtención de ambas predicciones implica un enorme gasto computacional.

Uno de los objetivos específicos autoimpuestos, es el diseño de algoritmos con tal grado de abstracción que se permita su implementación en diversas plataformas hardware como pueden ser las basadas en GPU, FPGA, microprocesador o soluciones que combinen varios tipos de dispositivos hardware. Un análisis de la imagen previo a la codificación permite extraer características del vídeo a procesar, que posibilitan llevar a cabo optimizaciones en la codificación teniendo en cuenta el funcionamiento del sistema visual humano, destinando el mayor esfuerzo y las tareas computacionalmente más intensivas a las zonas de la imagen en las que se presta mayor atención. Como la etapa de pre-análisis se ejecuta sobre la imagen de entrada, este proceso se puede independizar fácilmente de la codificación, permitiendo que el análisis sea ejecutado en el dispositivo encargado de llevar a cabo la codificación propiamente dicha o en otro dispositivo trabajando como co-procesador. A su vez, el análisis puede ejecutarse a nivel de imagen completa o trabajando por zonas, lo que posibilita que el proceso se ejecute en una etapa previa a la codificación o en el propio bucle de codificación.

Para permitir que los algoritmos puedan ser implementados en diferentes dispositivos hardware, se ha optado por el uso de un lenguaje que permite interoperabilidad entre diversos dispositivos y diferentes fabricantes. OpenCL es un lenguaje de programación para cómputo paralelo libre de regalías que cumple estos requisitos, por lo que los algoritmos de análisis y clasificación diseñados fueron implementados utilizando este lenguaje.

1.4.1 Optimizaciones basadas en la detección de regiones espacialmente homogéneas

Existen ciertas características presentes en la imagen que mediante su correcta interpretación, permiten llevar a cabo simplificaciones en el proceso de codificación sin incrementar considerablemente la tasa de datos generada y sin provocar una degradación visual perceptible. En el presente trabajo se ha llevado a cabo un análisis de la imagen a codificar con el objetivo de clasificar aquellas regiones que presentan una evidente homogeneidad espacial. La homogeneidad espacial puede deberse, según los algoritmos de clasificación implementados, a dos causas:

1. Presentar texturas homogéneas.
2. La existencia de una clara direccionalidad espacial predominante en los píxeles que la componen.

El primer grupo, es detectado mediante un análisis de la relación entre la energía de baja frecuencia respecto a la energía total. La clasificación se realiza a nivel de todos los tamaños de bloque que permite HEVC, por lo que se obtiene un mapa binario para cada tamaño posible. La aplicación de transformadas para la obtención de la energía de la imagen sería computacionalmente muy costoso, por lo que se ha optado por hacer uso de la Relación de Parseval para resolver este problema. De acuerdo a la Relación de Parseval, los dominios de la frecuencia y el tiempo son representaciones equivalentes de la misma señal y deben contener la misma energía. Por lo tanto, es posible trabajar en el dominio temporal a nivel de píxel calculando previamente la energía.

El segundo grupo es detectado tras aplicar sobre la imagen de entrada una predicción intra basada en el estándar H.264. La información extraída de la imagen es adaptada posteriormente al estándar HEVC teniendo en cuenta las similitudes y diferencias entre ambos estándares, para llevar a cabo en la codificación una serie de optimizaciones. La implementación de la predicción intra H.264 se ha llevado a cabo utilizando una extensión de OpenCL que invoca a módulos hardware de propósito dedicado existentes en las GPU de Intel. De esta forma, el coste computacional asociado a este proceso es prácticamente cero, liberando de esta tarea a las unidades lógicas de la GPU que son destinadas a otro propósito: la adaptación de los datos obtenidos y la clasificación de la imagen para obtener el correspondiente mapa binario.

Los mapas binarios obtenidos para cada grupo son compartidos con el procesador a cargo de la codificación propiamente dicha, en el que se llevan a cabo diferentes optimizaciones:

- Parada prematura en el proceso de división recursivo de la imagen. Si un bloque es clasificado como espacialmente homogéneo tras consultar el mapa binario para el tamaño correspondiente, se detiene el proceso recursivo.
- Reducción del número de modos de predicción intra a evaluar:
    - De 35 a un máximo de 5, para bloques que presentan texturas homogéneas.
    - De 35 a 9, para bloques con una direccionalidad espacial predominante.
- En la predicción inter, reducción del número de modos de partición a evaluar de 8 a únicamente 1, en bloques que presentan texturas homogéneas.

La implementación de estas optimizaciones dentro del bucle de codificación supone unas mínimas modificaciones en su código, por lo que pueden ser fácilmente implementadas en cualquier dispositivo. Las optimizaciones se aplican únicamente sobre regiones clasificadas como espacialmente homogéneas, por lo que pueden integrarse con los algoritmos propuestos en otros trabajos aplicados sobre el resto de la imagen.

1.4.2 Optimizaciones basadas en la detección de regiones temporalmente homogéneas

Una región temporalmente homogénea es aquella en las que los bloques que la constituyen presentan un movimiento muy similar o ningún movimiento (región estática). Al igual que en regiones espacialmente homogéneas, en las regiones temporalmente homogéneas se pueden realizar una serie de simplificaciones en el proceso de codificación afectando levemente a sus prestaciones.

Para detectar este tipo de regiones se requiere del uso de un algoritmo de estimación de movimiento aplicado sobre el vídeo a codificar, de manera que se pueda caracterizar el tipo de movimiento existente. La búsqueda de movimiento dentro de un vídeo, es un proceso computacionalmente muy elevado, por lo que en la aportación realizada se ha optado por el uso de una extensión de OpenCL que permite la utilización de un módulo de hardware dedicado presente en las GPU de Intel. De la estimación de movimiento se obtiene como resultado un conjunto de vectores de movimiento que caracterizan el movimiento existente en la imagen objeto de estudio. Al igual que la extensión de OpenCL utilizada para ejecutar la predicción intra, se trata de un módulo destinado a ser ejecutado en el ámbito del estándar H.264, por lo que existe un proceso posterior encargado de adaptar los datos obtenidos a HEVC y de realizar la clasificación. La métrica escogida para

realizar el cálculo de homogeneidad temporal presente, es la desviación media absoluta, ya que permite obtener una medida estadística de la dispersión existente en el movimiento dentro de un bloque de la imagen. Cuando el bloque es clasificado como temporalmente homogéneo, la estimación de movimiento realizada sobre la imagen de entrada a la codificación, permite describir de una manera bastante precisa el movimiento existente. Por lo tanto, la información de movimiento extraída puede ser aprovechada para simplificar el algoritmo de estimación de movimiento que se aplica en el cuerpo del codificador HEVC.

Los procesos de adaptación y clasificación se ejecutan en unidades lógicas de la GPU, y el mapa binario obtenido para cada tamaño de bloque permitido en HEVC es transferido al procesador a cargo de la codificación para llevar a cabo en los bloques clasificados como temporalmente homogéneos, las siguientes optimizaciones:

- Aplicación de un algoritmo optimizado de decisión de tamaño de bloque.
- Optimización del algoritmo de decisión de modo de partición inter.
- Implementación de un algoritmo de configuración adaptativa de la ventana de búsqueda en la estimación de movimiento.
- Parada prematura aplicada sobre la predicción inter, basada en la comparación de los costes obtenidos para la predicción intra e inter por los módulos hardware. La predicción inter puede llegar a no ejecutarse si el coste asociado a la predicción intra es considerablemente inferior.

Al tratarse de optimizaciones aplicadas sobre las regiones clasificadas como temporalmente homogéneas, los algoritmos propuestos pueden complementarse con la utilización de las optimizaciones aplicadas sobre las regiones espacialmente homogéneas y aplicando en el resto de la imagen, soluciones adoptadas en otros trabajos.

1.4.3 Mejoras perceptuales

A la hora de diseñar, implementar y caracterizar los algoritmos de optimización se ha tenido en cuenta el funcionamiento del sistema visual humano. Para tal propósito, se han utilizado una serie de métricas que incorporan en su cálculo consideraciones perceptuales.

Por otro lado, se han implementado algoritmos específicos para mejorar la experiencia del usuario que visualiza el vídeo descodificado por medio de un incremento de la calidad visual percibida:

- Modificación del valor del parámetro de cuantificación en regiones con texturas homogéneas.
- Eliminación de información redundante en regiones de la imagen que presentan una estructura caótica.
- Algoritmo de reducción de artefactos intrínsecos al HEVC.

El valor del parámetro de la cuantificación que se aplica tras la transformación en el proceso de codificación, influye directamente en la tasa de datos generada y en la calidad visual. Cuanto menor sea su valor, mayor calidad visual se obtiene a expensas de mayor consumo de bits. En la transmisión de vídeo a través de un canal de transmisión se requiere que la tasa de bits generada en un segundo se mantenga constante para asegurar que no se sobrepasa el ancho de banda del canal. El algoritmo encargado de mantener la tasa de datos

constante, se denomina algoritmo de control de tasa, y se encarga de establecer un presupuesto de bits para cada imagen y/o bloque de la imagen y de variar el valor del parámetro de cuantificación para asegurar que dicho presupuesto se cumpla. El ojo humano es más sensible a percibir artefactos debidos a la codificación en regiones con texturas homogéneas, por ello se utiliza la clasificación de texturas implementada en la etapa de pre-análisis para indicar al algoritmo de control de tasa que debe reducir el valor del parámetro de cuantificación en estas regiones. Como la reducción del valor del parámetro de cuantificación supone un incremento en el número de bits generado, la modificación realizada conlleva que el algoritmo de control de tasa aumente el valor del parámetro de cuantificación en el resto de regiones de la imagen con el objetivo de mantener la tasa de datos constante. Esta modificación del valor de cuantificación mejora considerablemente la percepción de calidad visual en la secuencia completa, ya que la calidad ha sido aumentada en aquellas regiones que son más susceptibles a ser foco de atención mientras que la pérdida de calidad sufrida en otras regiones de la imagen resulta imperceptible. De manera análoga, se puede introducir mayor cantidad de distorsión en las regiones de la imagen que presentan una estructura caótica antes de que dicha distorsión sea perceptible, debido a que el sistema visual humano no es capaz de percibir toda la información de la imagen relativa a altas frecuencias. Por este motivo, se ha implementado un algoritmo que realiza un filtrado en aquellas regiones que presentan una estructura caótica, lo cual reduce el detalle de dichas regiones, lo que directamente implica una reducción en el número de bits generado. Esta reducción permite que el algoritmo de control de tasa destine ese número de bits a otras regiones de la imagen para mejorar la calidad global percibida. La detección de las regiones con estructura caótica hace uso de la clasificación de texturas homogéneas y de un algoritmo basado en gradientes. El uso de la amplitud del gradiente y del ángulo asociado es utilizado en diversos trabajos para optimizar el algoritmo de decisión de tamaño de bloque y de modo de predicción intra. Por ello, se decidió emplear la amplitud del gradiente en la detección, ya que contribuye a la detección de regiones caóticas y permite además, que la información relativa al gradiente sea reaprovechada para llevar a cabo optimizaciones adicionales que reduzcan el gasto computacional asociado a la codificación HEVC. Esta asunción ha sido verificada en los experimentos realizados tras integrar un algoritmo optimizado de decisión de tamaño de bloque basado en gradientes con la solución adoptada. El algoritmo basado en gradientes es ejecutado sobre regiones que no han sido clasificadas por la etapa de pre-análisis propuesta, por lo que son soluciones perfectamente complementarias.

La utilización de un tamaño de bloque que abarque regiones con texturas homogéneas y zonas de la imagen con cierto grado de detalle produce artefactos intrínsecos a la codificación. El efecto que se aprecia es que en el residuo, se genera ruido en la zona con textura homogénea procedente de la zona que contiene detalle debido al hecho de aplicar una transformada que engloba en su interior regiones de la imagen que contienen coeficientes pertenecientes a baja frecuencia (texturas homogéneas) y altas frecuencias (bordes de la imagen y regiones con detalle). En el trabajo realizado se presenta un algoritmo de selección de tamaño de bloque que soluciona este problema, detectando los bloques propensos a producir artefactos mediante la consulta del mapa binario de regiones que presentan texturas homogéneas. Si un bloque presenta texturas homogéneas junto a otro tipo de texturas en su interior, se fuerza su división en bloques de tamaño inferior.

1.4.4 Verificación en tiempo real

El software de referencia HM está orientado a la evaluación de las distintas técnicas que componen el estándar y es utilizado dentro de la literatura relacionada con la optimización del proceso de codificación HEVC para evaluar los resultados obtenidos. Utilizando una configuración predefinida y un mismo software, resulta sencillo establecer una comparativa con otros trabajos previamente publicados en términos de tasa de datos generada, tiempos de codificación obtenidos y calidad visual. El principal problema que presenta este software, como ya se ha comentado, es que no se encuentra optimizado, lo que implica que los tiempos obtenidos en la codificación disten enormemente de los necesarios para posibilitar una codificación en tiempo real. Como la mayoría de publicaciones utilizan este software, existe un gran vacío en la literatura relacionada con algoritmos de optimización cuya implementación sea realmente viable en sistemas de codificación en tiempo real. En los experimentos realizados se ha verificado que los tiempos dedicados a ejecutar determinados algoritmos de optimización son superiores a la reducción que provocan en el tiempo de codificación en un sistema ejecutado en tiempo real. Por estos motivos, se han implementado todos los algoritmos presentados en un codificador destinado a ser ejecutado en tiempo real, verificando la idoneidad del diseño realizado para dispositivos de potencia de cómputo reducida, bajo coste y consumo.

Para poder establecer una comparativa con trabajos existentes en la literatura relacionada, los algoritmos también han sido implementados en el software de referencia, comprobando que las soluciones presentadas se encuentran dentro de los trabajos que ofrecen mejor relación entre la reducción de tiempos de codificación que provocan, el incremento de la tasa de datos generada y la pérdida de calidad visual sufrida, tal y como se muestra en el apartado de resultados.

1.4.5 Captura de vídeo en tiempo real

En todo el trabajo expuesto, los experimentos se realizaron empleando como señal de vídeo de entrada un fichero, pero los sistemas de codificación que están implementados en la mayoría de equipos profesionales utilizan como entrada una señal de vídeo digital generada en tiempo real que proviene de un reproductor multimedia o una cámara de vídeo. No existen trabajos que describan cómo implementar un sistema de captura de vídeo en tiempo real, por lo que se decidió describir los aspectos más relevantes del diseño de una aplicación que permitiese la captura de una señal de vídeo externa al sistema y su transmisión al procesador a cargo de la codificación. El sistema de captura diseñado se basa en el uso de una FPGA que se encarga de extraer el audio y el vídeo de un interfaz SDI de entrada. Se ha optado por el uso de este interfaz al ser el más utilizado en el mercado profesional de distribución y contribución de contenidos multimedia. El vídeo y audio extraídos de la señal de entrada son enviados a la GPU encargada de realizar el análisis de la imagen, la clasificación de las diferentes regiones de la imagen, y algoritmos de pre-procesado de la imagen. Los datos generados por la GPU son compartidos con el microprocesador en el que se realiza la codificación HEVC y se implementan los algoritmos de optimización diseñados. La gestión de la memoria realizada en la GPU y el microprocesador, se ha realizado de tal forma que no se requieren copias, la memoria es compartida entre ambos dispositivos. La comunicación entre la FPGA y la GPU se realiza a través de un bus PCIe. En la memoria se recomienda la lectura de diversos documentos y se describen las modificaciones más relevantes a realizar sobre diseños de referencia

existentes para implementar en FPGA el interfaz SDI y PCIe, haciendo especial hincapié en la parte de generación de los relojes necesarios. Además, se describe el diseño realizado para la extracción del vídeo y audio, almacenamiento en memoria interna y su posterior envío a través del bus PCIe, así como el protocolo de comunicación establecido con el driver del host y algunas funcionalidades adicionales que hacen que el sistema de codificación sea robusto ante la desconexión repentina de la señal de vídeo de entrada o un cambio en el formato de vídeo. Las interrupciones generadas por la FPGA cada vez que finaliza la transmisión de una imagen son utilizadas para sincronizar el sistema de codificación completo, las funcionalidades implementadas permiten que no se produzcan errores en la sincronización aunque se produzcan alteraciones de la señal de entrada de vídeo. Todo ello sin hacer uso de memoria externa dedicada en la FPGA, lo que reduce el retardo asociado al dispositivo de entrada y el coste del sistema. El diseño actúa como máster del bus PCIe, de modo que la FPGA puede iniciar transacciones a la memoria del sistema de manera autónoma, liberando al microprocesador de esta tarea. Consultando la literatura existente no se ha conseguido encontrar ningún trabajo donde se comente nada al respecto de este tipo de funcionalidades.

1.5 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo:

- Capítulo 2. Compresión de vídeo. Este capítulo se centra en la descripción de una serie de conceptos básicos relacionados con la señal de vídeo digital. Estos conceptos serán utilizados en apartados posteriores y permitirán comprender en detalle la necesidad de la compresión de vídeo en sistemas multimedia. Además, se definen los módulos que componen un sistema de compresión de vídeo y se detallan las métricas y el entorno de test que permiten caracterizar la eficiencia de la compresión de vídeo y la calidad de la señal descodificada.
- Capítulo 3. Estándares de compresión de vídeo. Se muestra la evolución de los diferentes estándares de compresión de vídeo a lo largo de la historia y sus principales aportaciones respecto a sus predecesores. A continuación se describen los aspectos fundamentales de la compresión de vídeo que sirven como base de conocimiento para comprender las nuevas técnicas que componen el estándar HEVC.
- Capítulo 4. Introducción al estándar HEVC. Se describe la arquitectura de un codificador HEVC y los módulos que lo componen para implementar las diferentes técnicas y herramientas. Se describen las principales diferencias respecto a H.264 y el capítulo finaliza con un apartado destinado a recalcar los principales inconvenientes que supone el uso de HEVC y las alternativas que existen actualmente y las que habrá en un futuro muy cercano.
- Capítulo 5. Contribuciones (codificación HEVC). Se exponen los diferentes algoritmos de optimización que han permitido acelerar el proceso de codificación y mejorar la calidad visual percibida.
- Capítulo 6. Contribuciones (captura de vídeo). Se describen los interfaces utilizados, el ancho de banda requerido por el sistema de codificación y ciertos

detalles relativos a la implementación del sistema, así como la arquitectura hardware finalmente escogida.

- Capítulo 7. Conclusiones. Se señalan los principios y relaciones que indican los resultados. Se relacionan los resultados obtenidos respecto trabajos existentes en la literatura y se establecen futuras líneas de trabajo.
- Apéndice 1. Se describen ciertos registros de control de la FPGA encargada de la captura de vídeo para poder establecer un protocolo de control y de comunicación con el driver del host.

1.6 Publicaciones

- D. G. Fernández, G. Botella, A. A. Del Barrio, C. García, M. Prieto, C. Grecos, "HEVC optimization based on human perception for real-time environments", in Multimedia Tools and Applications, December 2018. JCR-Q2.
- D. G. Fernández, A. A. Del Barrio, G Botella and C. García. "Fast and effective CU size decision based on spatial and temporal homogeneity detection", in Multimedia Tools and Applications, Volume 77, Issue 5, pp 5907–5927, March 2018. JCR-Q2.
- D.G. Fernández, A.A. Del Barrio, G. Botella, C. García, M. Prieto, R. Hermida. "Complexity reduction in the HEVC/H265 standard based on smooth region classification", in Digital Signal Processing, Volume 73, pp 24-39, February 2018. JCR-Q2.
- D. G. Fernández, A. A. Del Barrio, Guillermo Botella, Uwe Meyer-Baese, Anke Meyer-Baese, Christos Grecos, "Information fusion based techniques for HEVC", in Proc. SPIE Real-Time Image and Video Processing 2017, May 2017.
- D. G. Fernández, A. A. Del Barrio, Guillermo Botella, Uwe Meyer-Baese, Anke Meyer-Baese, Christos Grecos, "Pre-processing techniques to improve HEVC subjective quality", in Proc. SPIE Real-Time Image and Video Processing 2017, May 2017.
- D. G. Fernández, A. A. Del Barrio, G Botella and C. García. "Fast CU size decision based on temporal homogeneity detection", in Proc. Conference on Design of Circuits and Integrated Systems (DCIS) 2016, November 2016.
- D. G. Fernández, A. A. Del Barrio, G. Botella and C. García, U. Meyer-Baese, A. Meyer-Baese, "HEVC optimizations for medical environments", in Proc. SPIE Sensing and Analysis Technologies for Biomedical and Cognitive Applications 2016, Volume 98710B, May 2016.
- D. G. Fernandez, A. A. Del Barrio, G. Botella and C. Garcia, "4K-based intra and inter prediction techniques for HEVC", in Proc. SPIE Real-Time Image and Video Processing 2016, Volume 98970B, April 2016.

Capítulo 2

Compresión de vídeo

2.1 Introducción

El presente capítulo se centra en la descripción de una serie de conceptos básicos relacionados con la señal de vídeo digital que serán utilizados en apartados posteriores y que permitirán comprender en detalle la necesidad de la compresión de vídeo en sistemas multimedia.

El apartado 2.2 se centra en una breve definición de los conceptos relacionados con la señal de vídeo, en el apartado 2.3 se define la motivación que se esconde tras la compresión de vídeo, en el apartado 2.4 se explican muy brevemente los diferentes módulos que componen un sistema de compresión de vídeo y en el apartado 2.5, finalmente se detallarán las métricas y el entorno de test que permiten caracterizar la eficiencia de la compresión de vídeo y la calidad de la señal.

2.2 Conceptos relacionados con la señal de vídeo digital

En la actualidad existe una tendencia a aumentar la resolución de la imagen y el número de imágenes por segundo que los dispositivos de visualización (displays, monitores, televisores, etc.) pueden soportar, buscando mejorar la calidad del vídeo mostrado. A continuación, se procederá a la definir los conceptos más relevantes que permiten caracterizar una señal de vídeo.

2.2.1 Resolución

La resolución indica el número de píxeles en cada dimensión que son utilizados para representar una imagen, obviamente cuanto mayor número de píxeles se utilice, mayor detalle en la imagen se puede apreciar.

2.2.2 Espacio de color

Otro concepto importante relacionado con vídeo digital, es el espacio de color utilizado para almacenar una imagen. El espacio de color es una representación matemática de un conjunto de colores [1]. Existen distintos espacios de color, siendo los más populares RGB, YCbCr y CMYK. RGB es normalmente utilizado en la gestión de gráficos por ordenador, YCbCr en sistemas de compresión y procesamiento de vídeo y CMYK para impresión de color. Este último no se definirá, al estar su aplicación fuera del marco de trabajo de la presente Tesis.

2.2.2.1 RGB

El espacio de color RGB es ampliamente utilizado porque utiliza las componentes de color rojo, verde y azul que coinciden con las que utilizan los monitores para crear cualquier color. Por otro lado, para modificar la intensidad o color de un píxel concreto se requiere la modificación de los tres valores RGB, lo que en términos de cómputo exige mayor esfuerzo que si sólo se requiriese la modificación de una única componente, puesto que es necesario mayor número de accesos a memoria y realizar mayor número de operaciones.

2.2.2.2 YCbCr

Debido al mayor gasto computacional asociado al uso de RGB, el espacio de color YCbCr es el que normalmente es utilizado en compresión de vídeo. Como la presente Tesis está orientada a la compresión de vídeo digital nos centraremos en el uso del espacio de color YCbCr. En este espacio de color cada píxel es representado mediante tres componentes: Y (luminancia), Cb (crominancia azul) y Cr (crominancia roja).

Existen diversas fórmulas ampliamente conocidas que permiten la adaptación de un espacio de color a otro. Para el caso que tratamos, las Ecuaciones 2.1 y 2.2, permiten la conversión del espacio de color RGB a YCbCr y viceversa.

$$Y = k_r R + (1 - k_b - k_r)G + k_b B, \tag{2.1}$$

$$C_b = \frac{0.5}{1-k_b}\,(B - Y),$$

$$C_r = \frac{0.5}{1-k_r}\,(R - Y).$$

$$R = Y + \frac{1-k_r}{0.5} C_r, \tag{2.2}$$

$$G = Y - \frac{2k_b(1-k_b)}{1-k_b-k_r} C_b - \frac{2k_r(1-k_r)}{1-k_b-k_r} C_r,$$

$$B = Y + \frac{1-k_b}{0.5} C_b.$$

Los valores de las constantes k_r y k_b están definidos en distintas recomendaciones de la Unión Internacional de Telecomunicaciones (*International Telecommunication Union, ITU*) en función del tipo de señal de vídeo digital que se vaya a utilizar. El documento *Recommendation ITU-R BT.601-7* [2] define estas constantes para la televisión digital de definición estándar con los valores $k_r = 0.299$ y $k_b = 0.144$, y el documento *Recommendation ITU-R BT.709-6* [3] los define como $k_r = 0.2126$ y $k_b = 0.0722$ para televisión digital en alta definición. La luminancia (brillo) como se puede observar en la Ecuación 2.1, se calcula como una media pondera de las tres componentes RGB. Cada píxel siempre estará definido por al menos una componente de luminancia Y, y el número de muestras de crominancia viene determinado por el muestreo de color utilizado, que es definido en el siguiente apartado.

2.2.3 Muestreo de color (YCbCr)

Dentro del espacio de color YCbCr existen diferentes formatos de muestreo en función del número de muestras de crominancia que se utilicen para representar cada píxel, siendo los posibles formatos de muestreo 4:4:4, 4:2:2, 4:1:1 y 4:2:0.
En el caso de 4:4:4, cada píxel está representado por una muestra de luminancia Y, una muestra de crominancia Cb y una muestra de crominancia Cr. La Figura 2.1, muestra gráficamente este formato.

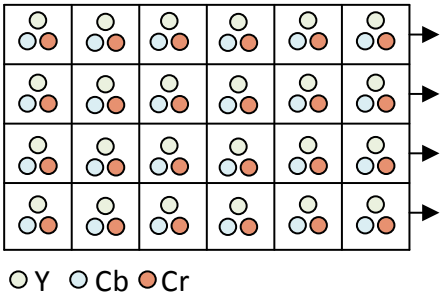


Figura 2.1. Muestreo de color 4:4:4.

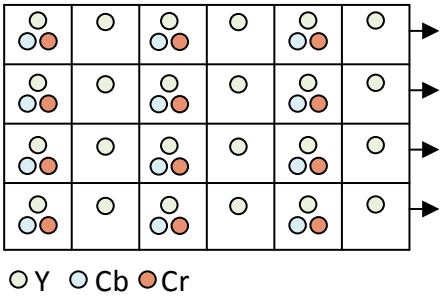


Figura 2.2. Muestreo de color 4:2:2.

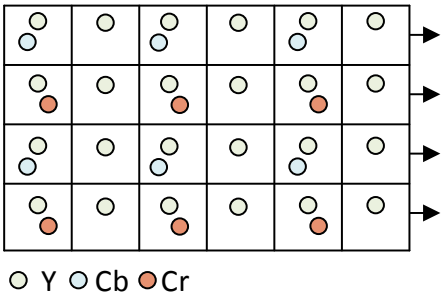


Figura 2.3. Muestreo de color 4:2:0.

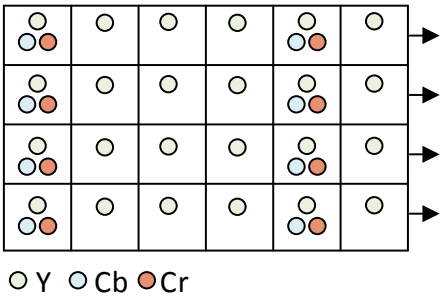


Figura 2.4. Muestreo de color 4:1:1.

Las figuras 2.2, 2.3 y 2.4 muestran los formatos de muestreo de color 4:2:2, 4:2:0 y 4:1:1, respectivamente. Normalmente, la luminancia es muestreada a mayor frecuencia que la crominancia debido a que el ojo es más sensible a las variaciones en el brillo que en la tonalidad. Existe la posibilidad de usar sistemas monocromáticos (4:0:0) pero no se puede considerar como un muestreo de color propiamente dicho, ya que sólo existen muestras de luminancia Y.

2.2.3.1 Formatos de píxel YCbCr

En función de cómo se ordenan en el espacio de memoria las muestras de luminancia y de crominancia que definen un píxel, existen gran cantidad de formatos de píxel. Dentro de los más utilizados en la compresión de vídeo destacan 4:2:2/4:2:0 *YUV planar* y 4:2:2 *UYVY interleaved*. Los términos YUV e YCbCr son intercambiables, por lo que a la componente de color azul se la puede denominar U o Cb y a la de color rojo, V o Cr. En los formatos denominados como planar, se almacenan todas las muestras de luminancia en un mismo búfer, las de color azul se almacenan todas a continuación de la luminancia, y las de color rojo después de las de color azul. El tamaño de la memoria destinada a cada espacio de color lo define el muestreo de color, ya que por ejemplo 4:2:2, contiene el doble de muestras de crominancia que 4:2:0. Por el contrario, en los formatos interleaved las muestras de luminancia y de crominancia van intercaladas dentro del mismo espacio de memoria. Para el caso de 4:2:2 UYVY interleaved, las muestras siguen el orden U0, Y0, V0, Y1, U2, Y2, V2, Y3, etc. El conjunto de muestras U0Y0V0Y1, denominado macro píxel, definiría los píxeles 0 y 1 de la imagen.

2.2.4 Precisión del píxel

Cada píxel puede representarse con diferente número de bits (8, 10, 12 o 16 bits) aportando la mayor precisión una mejor definición en la imagen, al ser capaz cada píxel de representar un mayor rango de valores tanto de luminancia como de crominancia, lo que aporta una mayor paleta de colores que se pueden representar.

2.2.5 Tasa de imágenes

Cuando se trabaja con vídeo digital, también es necesario conocer el número de imágenes por segundo (*framerate, fps*) que se utiliza, puesto que un vídeo es un conjunto de imágenes concatenadas. A mayor número de imágenes por segundo, mayor información proporcionaremos al cerebro sobre el movimiento que se produce en una secuencia de vídeo. El hecho de utilizar una tasa de imágenes reducida provoca que el movimiento que percibe el usuario no sea fluido.

2.2.6 Tipo de escaneo

Desde el punto de vista de la cámara, un vídeo es un conjunto de imágenes capturadas a una frecuencia de muestreo fija (igual al framerate). La señal de vídeo puede ser muestreada de dos formas distintas. En progresivo, utilizando imágenes completas (*frames*), o en entrelazado, capturando cada imagen mediante la unión de dos campos (*fields*) muestreados en dos instantes de tiempo distintos. En entrelazado cada campo contiene la información temporal de la mitad de la imagen, uno de los campos está conformado por las líneas de la imagen pares (*top field*) y el segundo, por las líneas impares de la imagen (*bottom field*). El formato entrelazado presenta la ventaja de permitir enviar más información de movimiento para una determinada frecuencia de muestreo. Por ejemplo, en el caso de utilizar 25 imágenes por segundo, la tasa de campos por segundo sería de 50, lo que puede permitir captar el movimiento de una determinada secuencia de

vídeo de una manera más precisa, dando la sensación al visualizar la secuencia de tener un movimiento más fluido.

2.2.7 Formatos de vídeo

La combinación de la resolución y el número de imágenes por segundo determinan los diferentes formatos de vídeo. En la Tabla 2.1 se muestran los formatos de vídeo más empleados. En la Tabla 2.1, los formatos PAL y NTSC son conocidos como formatos de definición estándar (*Standard Definition, SD*) y el resto, como alta definición (*High Definition, HD*).

Tabla 2.1. Formatos de vídeo más utilizados (HD y SD).

Nomenclatura	Número de líneas	Número de píxeles/línea	Número de imágenes/segundo	Escaneo
1080p60	1080	1920	60	Progresivo
1080p59	1080	1920	59.94	Progresivo
1080p50	1080	1920	50	Progresivo
1080p30	1080	1920	30	Progresivo
1080p29	1080	1920	29.97	Progresivo
1080p25	1080	1920	25	Progresivo
720p60	720	1280	60	Progresivo
720p59	720	1280	59.94	Progresivo
720p50	720	1280	50	Progresivo
720p30	720	1280	30	Progresivo
720p29	720	1280	29.97	Progresivo
720p25	720	1280	25	Progresivo
1080i60	1080	1920	30	Entrelazado
1080i59	1080	1920	29.97	Entrelazado
1080i50	1080	1920	25	Entrelazado
PAL	576	720	25	Entrelazado
NTSC	480	720	29.97	Entrelazado

Además de estos formatos, recientemente ha aparecido la televisión conocida como *Ultra High Definition Television (UHD-TV)* gracias en parte a la utilización del nuevo estándar de codificación HEVC. En la Tabla 2.2, se pueden consultar algunos de los formatos conocidos como UHD.

Tabla 2.2. Formatos de vídeo más utilizados (UHD).

Nomenclatura	Número de líneas	Número de píxeles/línea	Número de Imágenes/segundo	Escaneo
UHDTV-1 / 4K	2160	3840	60	Progresivo
UHDTV-2 / 8K	4320	7680	60	Progresivo

2.2.8 Tiempo real y latencia

Para finalizar con la definición de los parámetros más importantes que caracterizan la señal de vídeo digital, se realizará una breve introducción al concepto de ejecución en tiempo real y a la latencia (o retardo).

La ejecución en tiempo real en sistemas de vídeo se puede definir como la capacidad de un sistema de procesar las imágenes de entrada en un tiempo de procesamiento inferior a la tasa de imágenes por segundo definida por el formato de vídeo con el que se esté trabajando. Por ejemplo, en el caso de un codificador de vídeo trabajando con un formato 1080p50, para alcanzar tiempo real tendría que ser capaz de procesar al menos 50 planos por segundo. Por lo tanto, para este formato, se deberá procesar un plano en menos de 20 ms.

En sistemas multimedia, además de la codificación existen otra serie de procesos que trabajan con las imágenes de entrada como pueden ser escalados para cambiar la resolución de la imagen, filtros para eliminar ruido, conversiones de espacios de color o del muestreo de color y un largo etcétera. Habiendo tantos procesos trabajando de forma conjunta, difícilmente se podrá conseguir que se realicen todos los procesos en tiempo real, por lo que resulta necesario introducir cierta latencia en el sistema. De esta manera, mientras cada proceso sea capaz de procesar de forma independiente una imagen dentro del tiempo requerido, se podrá cumplir el requerimiento de tiempo real. La introducción de latencia provoca que la salida del sistema tarde más tiempo en proporcionarse, siendo este tiempo la suma de todas las latencias de los distintos módulos que componen el sistema. En determinadas aplicaciones, el retardo introducido puede provocar una importante pérdida en la calidad de la experiencia vivida por el usuario, como pueden ser videoconferencias o contribuciones en directo de noticias donde no se pueden mantener comunicaciones fluidas entre los interlocutores.

2.2.9 Motivación

Una vez definidos los anteriores parámetros que permiten determinar el tamaño de un vídeo digital, se puede justificar fácilmente la necesidad que existe de utilizar la compresión de vídeo para posibilitar la transmisión del vídeo, en función del ancho de banda disponible en el canal de comunicación.

Suponiendo que se desea realizar la transmisión de un vídeo en tiempo real en formato 1080p50, el número de bits por segundo (*bitrate*) que se requieren, es fácilmente computable con los parámetros introducidos previamente:

Resolución de una imagen: 1920 x 1080 píxeles,
Precisión de cada píxel: 8 bits,
Número de imágenes por segundo: 50,
Espacio de color: YCbCr,
Formato de muestreo de color: 4:2:2 (N muestras Y, N/2 muestras Cb y N/2 muestras Cr, siendo 'N' la resolución de una imagen),

Número de bits por segundo = 1920 x 1080 x 8 x 50 x 2 = 1.658.880.000 bits/s
≈ 1,659 Gbps.

Como se puede observar la posibilidad de obtener un canal de estas características se limita a la utilización de ciertas tecnologías como pueden ser los enlaces de fibra óptica, siendo totalmente inabordable para tecnologías ADSL, 3G/4G y enlaces por satélite, entre otras. El uso de un ancho de banda mayor supone un incremento en el coste de la transmisión, por lo que la inclusión de la compresión de vídeo en un sistema de transmisión de vídeo resulta necesaria para la amplia mayoría de aplicaciones. Otras aplicaciones, como el almacenamiento de vídeos digitales para su posterior archivado se benefician de una reducción del tamaño del vídeo directamente en una reducción de la capacidad de almacenamiento requerida, lo que también repercute en el coste del sistema.

Actualmente, crece el ancho de banda disponible en las conexiones a través de Internet a la vez que se reduce su coste, por lo que en este escenario podría no resultar tan obvia la necesidad de la compresión de vídeo. Sin embargo, la utilización de la compresión de vídeo incluso utilizando tasas de datos muy elevadas supone un enorme beneficio, puesto que permite usando el mismo ancho de banda, poder transmitir imágenes de mayor resolución y mejor calidad visual, mejorando considerablemente la experiencia vivida por el usuario.

2.3 Sistemas de compresión de vídeo

El objetivo del anterior apartado es explicar la necesidad del uso de la compresión de vídeo para ahorrar costes en sistemas multimedia y/o para mejorar la calidad visual percibida por el usuario final sobre un sistema ya existente. En este apartado se explicará brevemente los diferentes elementos que componen un sistema de compresión de vídeo. El esquema básico es mostrado en la Figura 2.5.

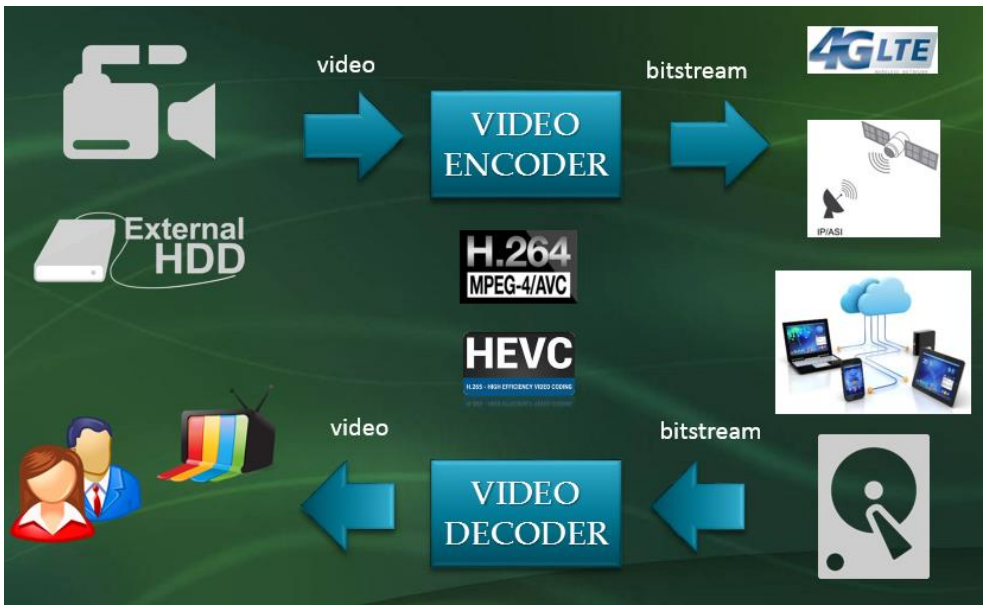


Figura 2.5. Esquema básico de un sistema compresión de vídeo.

El primer componente es la fuente de vídeo digital sin comprimir que proporciona una entrada al sistema. Esta fuente puede ser una cámara, un disco duro multimedia, un

generador de señales o cualquier otro dispositivo que proporcione una señal de vídeo directamente al sistema. El vídeo a codificar también puede provenir de un fichero previamente almacenado. La señal procedente de la fuente de vídeo puede tener que ser procesada para adaptarla a lo que se requiere a la entrada del codificador. Este pre-procesado de la imagen puede contener conversiones de espacios de color (por ejemplo, RGB a YCbCr), conversiones entre diferentes tipos de muestreo de color (p. ej. 422 a 420), escalados (p. ej. conversión de formatos SD a HD y viceversa), filtrados para eliminación de ruidos presentes en la señal, cambios de brillo o contraste y un amplio etcétera.

Cada imagen que compone el vídeo pre-procesado será posteriormente convertida a un conjunto de bits que conforman finalmente el *bitstream* que se obtiene como resultado tras aplicar la compresión de vídeo en el codificador. Dicho bitstream, puede ser encapsulado para ser transmitido a través de un canal de comunicación o ser tratado como un fichero para su almacenamiento.

El bitstream posee una sintaxis que es definida en el estándar de compresión de vídeo que se esté utilizando (MPEG-2, H.264, HEVC, etc.). Además de la sintaxis, el estándar define de forma inequívoca el proceso a seguir durante la descodificación para poder obtener las imágenes que componen el vídeo descodificado a partir del bitstream. Este vídeo descodificado puede ser posteriormente tratado en una etapa de post-procesado que puede incluir escalados en la resolución de la imagen, conversiones en la tasa de imágenes por segundo o espacios de color para adaptarse al monitor que se utiliza para la visualización, cambios del tipo de muestreo de color, reducción de artefactos debidos al proceso de codificación, etc.

Es importante resaltar que es en el módulo codificador de vídeo donde se define la eficiencia en la compresión y la calidad visual obtenida en el sistema en función de las diferentes técnicas que se empleen durante el proceso de compresión, no estando este proceso estandarizado. El estándar de vídeo proporciona una serie de técnicas que están disponibles, pero cómo implementarlas es una ardua labor a realizar por la persona a cargo del diseño del sistema. Por ello, es en el campo de la codificación donde se centran los mayores esfuerzos de investigación, destinados a obtener las mayores tasas de compresión de datos, sin pérdida apreciable en la calidad visual y con el menor coste computacional posible.

2.4 Entorno de pruebas

2.4.1 Métricas Objetivas

Para poder evaluar las prestaciones de las modificaciones que se realicen respecto a un software codificador de referencia o respecto a otros trabajos publicados en la literatura existente, es necesario el uso de una serie de métricas que permitan caracterizar los algoritmos implementados de una manera objetiva. Dicha objetividad se consigue mediante métricas que permitan obtener una repetitividad en las pruebas, es decir obtener los mismos resultados cuando no se varían las condiciones de test. Dentro del estado del arte relacionado con la codificación de vídeo existen una serie de métricas ampliamente difundidas como son la variación en el tiempo de ejecución, el incremento o reducción en el tamaño del bitstream generado y para conocer una estimación de la calidad visual

aportada, la métrica *Peak Signal-to-Noise Ratio (PSNR)* [4-7]. PSNR es el ratio entre el valor de potencia máximo de una señal y la potencia de ruido.

Como realmente lo que se desea realizar es una comparación con el codificador empleado como referencia, se trabaja con la diferencia respecto al caso base en términos de porcentajes, excepto para la PSNR, en la que se realiza una resta entre el valor obtenido por el codificador modificado y el codificador de referencia. Por lo tanto, las métricas normalmente utilizadas son las que se muestran en las ecuaciones 2.3, 2.4 y 2.5.

$$\Delta\text{Bitstream}\ (\%) = \frac{Proposal\ (bytes) - Reference\ (bytes)}{Reference\ (bytes)} * 100. \quad (2.3)$$

$$\Delta\text{Time}\ (\%) = \frac{Proposal\ Time - Reference\ Time}{Reference\ Time} * 100. \quad (2.4)$$

$$\Delta PSNR\ (dB) = Proposal\ PSNR - Reference\ PSNR. \quad (2.5)$$

Como se puede derivar de las ecuaciones anteriores, un valor negativo significaría una reducción respecto al caso base o referencia.

La eficiencia en un codificador de vídeo se puede considerar como la capacidad que tiene de a partir de un vídeo de entrada, obtener un bitstream del menor tamaño posible, proporcionando una gran calidad visual y siendo el tiempo de ejecución reducido. Cuando se comparan diferentes codificadores, configuraciones o modificaciones de un mismo codificador; uno de los aspectos más importantes a tratar es la definición de un conjunto de condiciones de test comunes a todas las pruebas. Resulta esencial el uso de un banco de pruebas extenso en el que se empleen secuencias de vídeo diferentes resoluciones y distintos tipos de texturas y movimientos. Para este propósito y en el caso exclusivo de la codificación HEVC, la organización *Joint Collaborative Team on Vídeo Coding (JCT-VT)* recomienda un conjunto de secuencias de vídeo y unas condiciones de test comunes en el documento *Common test conditions and software reference configurations* [8]. Se trata de un documento muy útil para realizar comparaciones de prestaciones respecto a otros trabajos. Dentro de las principales novedades que presenta este documento, está la aparición como nueva métrica de caracterización de *Bjontegaard Delta Rate (BD-Rate)* [9]. Esta medida tiene en cuenta el tamaño del bitstream generado y la medida de calidad PSNR, por lo que permite reducir el número de métricas necesarias para caracterizar un codificador. Cuanto menor sea el valor de BD-Rate, más eficiente será el algoritmo, ya que implica una menor pérdida visual y un menor incremento en la tasa de datos generada. La segunda métrica que utilizan las condiciones de test propuestas en [8] es la variación en tiempo de ejecución que ya se comentó previamente. En [8] se establece como software de referencia o caso base, el modelo de test HM que se definirá en detalle en posteriores apartados. Entre las opciones de configuración que se proponen en [8], aparece el uso de únicamente cuatro valores del parámetro de cuantificación (*Quantization Parameter, QP*). Los valores escogidos son 22, 27, 32 y 37; lo que parece insuficiente para caracterizar adecuadamente a un algoritmo de codificación que en condiciones de funcionamiento reales puede utilizar todos los valores posibles de QP (0-51) para ajustar la tasa de transmisión a las condiciones de la red que se esté utilizando. En las condiciones comunes de test [8] se proponen como secuencias de vídeo de alta resolución únicamente cuatro secuencias 4K, tres 720p y cinco 1080p, por lo que se decidió incluir en las pruebas un mayor número de secuencias de vídeo HD para caracterizar con mayor detalle los algoritmos propuestos. Dichas secuencias adicionales fueron obtenidas de los repositorios de acceso libre [10], [11] y [12]. Por lo tanto, las opciones de test propuestas en [8] son de enorme utilidad para establecer un entorno de test que permita la comparación con otros

trabajos, pero dicha propuesta debe de ser complementada con nuevas configuraciones y secuencias para una detallada caracterización de los algoritmos.

Tabla 2.3. Secuencias de vídeo utilizadas.

Repositorio	Resolución	Secuencias
EBU and SVT [12]	4K	CrowdRun_3840x2160, DucksTakeOff_3840x2160, InToTree_3840x2160, OldTownCross_3840x2160, ParkJoy_3840x2160
Ultra Vídeo Group [11]	4K	Bosphorous, ShakeNDry, Coastguard, Jockey
JCT-VT [8]	3G	BasketballDrive_1920x1080p50, BQTerrace_1920x1080p60, Cactus_1920x1080p50
	1080p24	Kimono_1920x1080p24, ParkScene_1920x1080p24
	720p60	FourPeople_1280x720p60, Johnny_1280x720p60, KristenAndSara_1280x720p60
	832x480p	BasketballDrill_832x480p50, BQMall_832x480p60, PartyScene_832x480p50, RaceHorses_832x480p30
xiph.org [10]	3G	DucksTakeOff_1080p50, InToTree_1080p50, ParkJoy_1080p50, CrowdRun_1080p50
	1080p25/30	blue_sky_1080p25, pedestrian_area_1080p25, rush_hour_1080p25, tractor_1080p25, riverbed_1080p25, aspen_1080p30, sunflower_1080p25, snow_mountain_1080p30, life_1080p30,
	720p	mobilecalendar_720p50, parkrun_720p50, shields_720p59, stockholm_720p60

2.4.2 Consideraciones Perceptuales

Presenta cierta obviedad que la precisión visual obtenida a la salida del equipo descodificador debe ser medida para verificar las prestaciones de un sistema de compresión de vídeo. Para tal propósito, se ha usado ampliamente la métrica PSNR debido a que es fácilmente implementable, pero esta métrica no tiene en cuenta el sistema visual humano (*Human Visual System, HVS*) por lo que no representa fielmente la perspectiva del usuario que visualiza el vídeo. Para intentar paliar este problema han surgido varias métricas con consideraciones perceptuales que incluyen diferentes aproximaciones matemáticas para

emular el comportamiento del ojo humano. Dentro de estas métricas, las más utilizadas son las siguientes:

- *PSNR-HVSM* [13]. Esta métrica utiliza coeficientes de la transformada de coseno discreta (*Discrete Cosine Transform, DCT*) y la tabla de cuantificación de JPEG para integrar características del HVS en la PSNR. Además, incluye un modelo de enmascaramiento que es utilizado para emular el comportamiento del HVS.
- *Structural Similarity Index (SSIM)* [14]. Está basada en la asunción de que el ojo humano ha evolucionado para extraer información estructural. Por ello, las imágenes que se utilizarán en la comparativa son descompuestas en luminancia, contraste y estructura. Posteriormente, se realizada una comparación por parejas de cada una de las componentes y se obtiene un índice de similitud mediante una combinación de los resultados obtenidos en las comparaciones realizadas sobre las tres componentes.
- *Multi Scale SSIM (MS-SSIM)* [15]. Se basa en aplicar la métrica SSIM a varias versiones de la imagen que han sido escaladas para reducir su resolución. El resultado final es una media ponderada de los valores SSIM obtenidos.
- *Three-Component Weighted Structural Similarity Index (3-Component SSIM Index, 3-SSIM)* [16]. Para el cómputo de esta métrica la imagen es dividida en tres tipos de regiones: bordes, zonas con texturas y regiones homogéneas. A partir de la media ponderada obtenida tras aplicar la métrica SSIM a estos tres tipos de regiones se obtiene el resultado final.
- *Spatial-Temporal SSIM (ST-SSIM)* [17]. En esta métrica se aplica SSIM a varias ventanas con diferentes orientaciones para tener en cuenta las distorsiones temporales.
- *Visual Information Fidelity (VIF/VIFP)* [18]. VIF es una generalización de la métrica *Information Fidelity Criterion (IFC)* [19]. En IFC, la evaluación de la calidad es modelada como un canal de transmisión. La información mutua entre la entrada y la salida a este canal se utiliza para cuantificar la información que el HVS puede percibir de la imagen bajo test. VIF utiliza la misma estructura comparando la información mutua entre las imágenes de salida del canal de transmisión con una versión de las mismas imágenes después de añadir distorsión al canal de transmisión. VIFP, es una versión de VIF trabajando en el dominio de los píxeles para reducir el coste computacional de la métrica.
- *Vídeo Quality Metric (VQM)* [20]. Está basada en una versión simplificada del modelo espacio-temporal de sensibilidad de contraste del ojo humano aplicando la DCT sobre las imágenes de entrada al sistema.

2.4.3 Métricas subjetivas

Mediante las métricas previamente definidas se puede obtener de una manera objetiva unos resultados repetitivos, pero los resultados obtenidos todavía distan de representar con certeza la opinión de un ser humano. La medida de calidad visual de un vídeo es intrínsecamente subjetiva, debido a que el sistema visual humano así lo es. La opinión referente a la calidad de una secuencia de vídeo depende de muchos factores como pueden ser la distancia a la pantalla, el estado de ánimo del usuario, su opinión personal sobre el contenido de la misma, cansancio, etc. Como se puede observar la mayoría de factores son inherentemente subjetivos, por lo que a las medidas que hacen uso de estadísticas basadas

en la opinión de distintas personas sobre la visualización de un mismo vídeo, se las denomina medidas subjetivas. La medida más difundida en la literatura de la compresión de vídeo es la métrica subjetiva *Mean Opinion Score (MOS)*, especificada en la norma ITU-R BT.500 [21], y más concretamente el método *Double Stimulus Impairment Scale (DSIS)*. La norma concreta las condiciones generales de observación (resolución del monitor, brillo y contraste, distancia de visualización, etc.). Dicho método consiste en presentar secuencialmente los vídeos en parejas: siendo el primero el vídeo original sin comprimir y el segundo la versión codificada. Después de la visualización de los vídeos por un número de expertos superior a 20, se realiza un cuestionario para que puntúen a la versión codificada acorde al siguiente criterio:

1. Muy molesto,
2. molesto,
3. ligeramente molesto,
4. perceptible, pero no molesto,
5. imperceptible.

En el trabajo presentado en [22] los autores presentan las métricas objetivas más utilizadas en la literatura y exponen cómo ninguna de ellas es lo suficientemente precisa para representar la opinión humana en comparación con MOS. Actualmente, éste es un campo de investigación abierto en el que se están continuamente publicando trabajos al respecto.

2.4.4 Problemática de las comparativas

Como se ha comentado con anterioridad, HEVC incluye numerosas herramientas nuevas para mejorar la eficiencia en la codificación. Los codificadores HEVC [23, 24] incluyen gran cantidad de parámetros de entrada que permiten habilitar o deshabilitar dichas herramientas. De esta forma, se puede establecer la configuración que proporcione mejor relación de compromiso (*trade-off*) en función de unos objetivos previamente fijados. Básicamente, se desea encontrar la configuración que proporcione mayor calidad visual, menor tasa de datos generada y todo ello, empleando el menor tiempo de ejecución posible. Menor gasto computacional permite el uso de dispositivos más baratos y reducir el consumo de potencia debido a que estos factores están directamente relacionados con la complejidad computacional del algoritmo. Una menor cantidad de datos generados permite la reducción de costes debido a la necesidad de un ancho de banda inferior o de dispositivos de almacenamiento de menor capacidad. La calidad visual es de vital importancia debido a que es el factor que proporciona una mejor experiencia desde la perspectiva del usuario final.

Para que un codificador pueda procesar vídeo en tiempo real se requiere de una profunda optimización de las instrucciones que componen su código, pero las mayores mejoras se obtienen reduciendo la complejidad de las diferentes herramientas y técnicas implicadas en la codificación. Cuando sobre una de estas técnicas se realiza una simplificación para reducir el tiempo de ejecución, se produce normalmente un incremento en la tasa de datos generada y una pérdida de calidad visual asociada, por lo que el diseñador del sistema tiene que analizar cuál es el mejor trade-off en función de sus intereses. Se trata de una ardua tarea debido a la gran cantidad de parámetros que se permiten variar en las diferentes configuraciones y en las posibles optimizaciones. Para facilitar la realización de esta tarea de una manera rápida y eficaz se decidió utilizar en el

análisis de resultados, el Óptimo de Pareto [25]. Se trata de un concepto que hace posible encontrar la configuración óptima que nos permita cumplir con los requerimientos de mayor reducción en tiempos de ejecución, imperceptible pérdida de calidad visual y menor incremento en tasa de datos generada.

Por otro lado, el Óptimo de Pareto también puede ser aplicado en una comparativa de distintos métodos para evaluar el trabajo realizado respecto a otros trabajos presentes en la literatura existente. Las mejores configuraciones o métodos son identificados como aquellos que corresponden a los puntos que pertenecen al frente de Pareto, siendo éstos los puntos que no son dominados por otros puntos para ciertos objetivos definidos por los ejes de coordenadas del gráfico. En el caso que nos ocupa, el análisis podría considerarse como un problema multi-variable y podría resolverse mediante dos gráficos. Un gráfico representaría el dominio variación en la tasa de datos generada respecto la variación en tiempo de ejecución (ΔBitstream /ΔTime). Mediante un segundo gráfico, se representaría el dominio variación de calidad visual respecto, al igual que en el caso anterior, variación en tiempo de ejecución (ΔPSNR/ΔTime). Gracias al uso de la métrica previamente mencionada BD-Rate [9], puede emplearse para el análisis un único gráfico, ya que esta medida tiene en cuenta la variación de calidad visual respecto la tasa de datos generada. Por lo tanto, un gráfico en el dominio BD-Rate frente a variación en el tiempo de ejecución sería suficiente para resolver el problema planteado, simplificando considerablemente el análisis. En la Figura 2.6, se muestra un ejemplo de comparativa de distintas configuraciones.

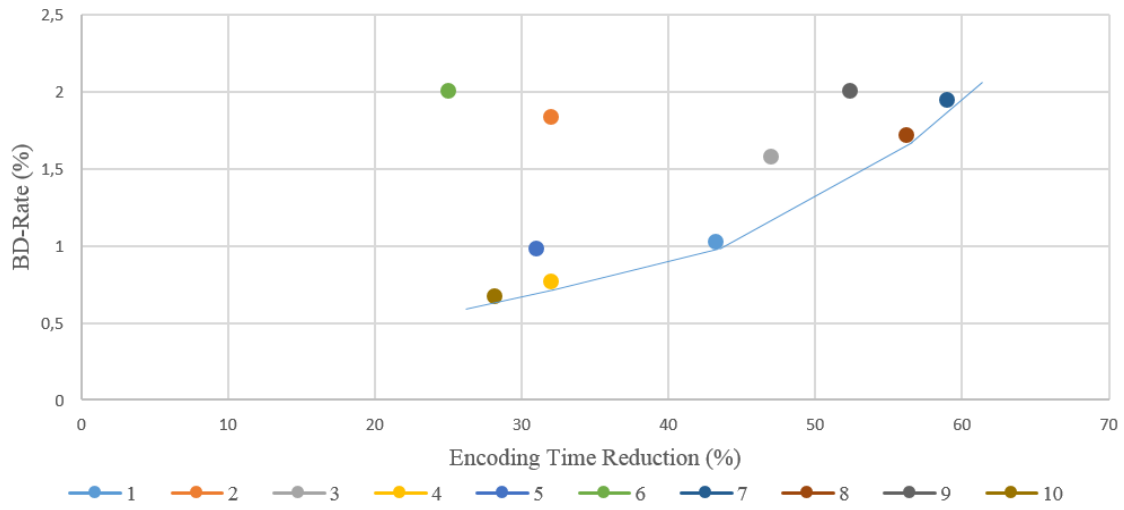


Figura 2.6. Análisis de Pareto en el dominio "ΔBD-Rate/ΔTime"

En función de que prime la reducción de la carga computacional o un menor BD-Rate, la configuración escogida puede ser una u otra. El uso del gráfico simplifica el problema de averiguar qué configuración proporciona la mejor relación de compromiso. En azul se indican los puntos pertenecientes al frente de Pareto. En cada punto indicado, para determinado valor de reducción de tiempo de ejecución no existe otro punto que ofrezca menor BD-Rate.

Para determinadas aplicaciones se requiere que el sistema tenga un consumo de potencia reducido, por lo que podría resultar necesario añadir este parámetro en el análisis. Otro importante factor que tiene relación directa con el éxito en el desarrollo de un proyecto es el precio del dispositivo de codificación y el hardware subyacente (velocidad y tipo de las memorias externas necesarias, periféricos, etc.), así que el coste del sistema podría ser

otro parámetro requerido. En este caso se podría hacer uso de dos gráficos, uno haciendo uso del espacio ΔBD-Rate/ΔTime y otro en el dominio Consumo de Potencia/Coste del Sistema, de manera que las configuraciones/métodos óptimos serían aquellos pertenecientes al frente de Pareto en ambos gráficos.

2.5 Referencias

[1] Jack, K. (2005). Video Demystified: A Handbook for the Digital Engineer (4ª Ed.). Oxford, UK: Elsevier.

[2] International Telecommunication Union. (March 2011). Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. Recommendation ITU-R BT.601-7.

[3] International Telecommunication Union. (June 2015). Parameter values for the HDTV standards for production and international programme exchange. Recommendation ITU-R BT.709-6,

[4] Wu D., Pan F., Lim K., Wu S., Li Z., Lin X., Rahardja S., Ko C. (2005). Fast intermode decision in h.264/AVC video coding. IEEE Transactions on Circuits and Systems for Video Technology 15(7):953–958.

[5] Shen L., Liu Z., Zhang Z., Shi X. (2008). Fast Inter Mode Decision Using Spatial Property of Motion Field. IEEE Transactions on Multimedia 10(6):1208–1214.

[6] Shen L., Liu Z., Liu S., Zhang Z., An P. (2009). Selective disparity estimation and variable size motion estimation based on motion homogeneity for multi-view coding. IEEE Transactions on Broadcasting 55(4).

[7] Shen L., Liu Z., Yan T., Zhang Z., An P. (2010). View-adaptive motion estimation and disparity estimation for low complexity multiview video coding. IEEE Transactions on Circuits and Systems for Video Technology 20(6).

[8] Bossen F. (2012). Common test conditions and software reference configurations. JCT-VC Document, JCTVC-K1100.

[9] Bjontegarrd G. (2001). Calculation of average PSNR differences between RD curves. ITU-T SC16/Q6 13th VCEG meeting, Austin.

[10] xiph.org. (Enero de 2016). Derf's test media collection. Recuperado de https://media.xiph.org/video/derf/

[11] Ultra Video group, Tampere University of Technology. (Enero de 2016). Test sequences. Recuperado de http://ultravídeo.cs.tut.fi/#testsequences

[12] European Broadcasting Union (EBU). (Enero de 2016). The EBU and SVT Public Test Sequences. Recuperado de https://tech.ebu.ch/testsequences

[13] Ponomarenko N., Silvestri F., Egiazarian K., Carli M., Astola J., Lukin V. (2007). On Between-Coefficient Contrast Masking of DCT Basis Functions. Paper presented at

Third International Workshop on Video Processing and Quality Metrics (VPQM), Scottsdale, United States.

[14] Wang Z., Bovik AC., Sheikh HR., Simoncelli EP. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing 13(4):600–612.

[15] Li C., Bovik AC. (2009). Three-component weighted structural similarity index. Proc. SPIE 7242–72420, Image Quality and System Performance VI. Paper presented at IS&T/SPIE Electronic Imaging, San Jose, California, United States.

[16] Moorthy AK., Bovik AC. (2010). Efficient motion weighted spatio-temporal video SSIM Index. Proc. SPIE 7527–75271. Paper presented at Human Vision and Electronic Imaging XV, San Jose, CA, United States.

[17] Sheikh HR., Bovik AC. (2006). Image Information and Visual Quality. IEEE Transactions on Image Processing 15(2):430–444.

[18] Sheikh HR., Bovik AC., de Veciana G. (2005). An Information Fidelity Criterion for Image Quality Assessment Using Natural Scene Statistics. IEEE Transactions on Image Processing 14(12):2117–2128.

[19] Xiao F. (2000). DCT-based Video Quality Evaluation—Final Project for EE392J.

[20] International Telecommunication Union. (January 2012). Methodology for the Subjective Assessment of the Quality of Television Pictures. ITU-R BT.500.

[21] Grazia M., Amadeo R. (2014). A new categorization of image quality metrics based on a model of human quality perception. World Acad. Sci. Eng. Technol. Int. J. Comput. Control Quantum Inf. Eng. 8 (6).

[22] McCann K., Rosewarne C., Bross B., Naccari M., Sharman K., Sullivan G. (2014). High efficiency video coding (HEVC) encoder description 0v16 (HM16). JCT-VC High Efficiency Video Coding N14 703.

[23] x265 Project. (Junio de 2019). Recuperado de http://x265.org/

[24] Smith, S. W. (1997). The Scientist and Engineer's Guide to Digital Signal Processing (2ª Ed.). California, USA: California Technical Publishing.

Capítulo 3

Estándares de Compresión de vídeo

3.1 Introducción

Las técnicas de compresión aplicadas al vídeo digital han contribuido enormemente al avance de las telecomunicaciones y sistemas multimedia, debido a que en estos ámbitos la optimización del uso del ancho de banda disponible siempre ha sido un factor determinante. Veinte años atrás se podía considerar la compresión de vídeo como algo novedoso, en la actualidad está presente en todo tipo de dispositivos. La compresión de vídeo ha permitido la implantación de la televisión digital terrestre (TDT), la visualización de películas en dispositivos móviles, la generalización de plataformas de contenidos bajo demanda, videoconferencias, etc. Existiendo tal cantidad de dispositivos haciendo uso de una tecnología, resulta necesaria la estandarización de los procesos implicados en la compresión de vídeo para permitir la interoperabilidad entre los distintos fabricantes.

Los estándares de compresión de vídeo son creados por diferentes organismos internacionales de estandarización de los que forman parte expertos del sector, incluyendo institutos de investigación, universidades y empresas privadas. Los dos principales organismos de estandarización relativos a la compresión de vídeo son *International Standards Organisation (ISO)* e ITU. El objetivo de la interoperabilidad es perseguido junto a la necesidad de incluir las últimas tecnologías de compresión de vídeo disponibles en ese momento, para que el estándar sea ampliamente adoptado por las ventajas que ofrezca respecto a estándares anteriores o soluciones propietarias alternativas. Por ello, antes de la creación de un nuevo estándar se establece un período inicial en el que los diferentes miembros proponen una serie de técnicas a implementar en el nuevo estándar. Dichas técnicas son evaluadas y discutidas en diferentes reuniones, estableciéndose una

competición entre las diferentes técnicas a adoptar. Dentro de los términos que determinan la elección de una propuesta, los más importantes son la calidad visual aportada y la tasa de datos generada, pero también la viabilidad de su implementación. Este último aspecto es fundamental para la implantación de un estándar, ya que procesos computacionalmente muy costosos pueden llevar a que el coste de los dispositivos utilizados sea tan elevado que se descarte el uso del nuevo estándar frente a otras alternativas.

Algunas de las técnicas adoptadas en los estándares son patentadas, lo cual lleva al uso de licencias asociadas, y a un negocio que puede llegar a ser muy lucrativo, por lo que numerosas empresas presentan diversas propuestas con el objetivo de recibir diferentes tipos de regalías. Por otro lado, participar activamente en el desarrollo del estándar permite a las empresas involucradas ir conociendo con antelación a su publicación las técnicas que lo constituyen, y de esta forma ir implementándolas de manera que cuando se produzca finalmente la publicación del estándar puedan proporcionar rápidamente productos al mercado; con la clara ventaja competitiva que esto ofrece.

La cobertura del estándar de compresión de vídeo no abarca al codificador. El estándar describe la sintaxis que debe seguir el conjunto de bits obtenidos tras la codificación, la semántica de los elementos de sintaxis y el proceso mediante el cual los elementos de sintaxis son descodificados para producir información visual perceptible. El estándar además define una serie de niveles (*levels*) y perfiles (*profiles*) que delimitan indirectamente la funcionalidad del codificador. Los niveles establecen los valores máximos de ciertos parámetros de funcionamiento de un descodificador, como por ejemplo, la tasa de bits por segundo que puede descodificar o la máxima resolución de la imagen. Los perfiles definen las herramientas/técnicas que se deben implementar.

3.2 Historia

La primera aproximación a la compresión se basa en un concepto obvio, aprovechando la redundancia existente en una señal, asignar los códigos de menor longitud a los símbolos con mayor probabilidad de ocurrencia y los de mayor longitud a los menos probables. Este tipo de codificador es conocido como codificador entrópico [1]. Los codificadores entrópicos obtienen buenas prestaciones cuando se trabaja con datos que presentan cierta independencia entre sí. En el caso de las secuencias de vídeo, donde se trabaja con píxeles que presentan un gran índice de correlación, se pueden obtener mayores niveles de compresión aprovechando las redundancias existentes. En el caso de una imagen, suele existir una gran correlación entre los píxeles vecinos por lo que existe una gran redundancia espacial. En un vídeo, las sucesivas imágenes que componen la secuencia suelen ser similares entre sí, por lo que tienden a presentar una alta redundancia temporal. Por otro lado, el HVS presenta ciertas limitaciones que pueden ser aprovechadas para eliminar toda aquella información que no vaya a ser percibida por nuestro ojo. Por ejemplo, HVS es más sensible a las bajas frecuencias que a las altas, por lo que se puede reducir considerablemente la cantidad de datos a procesar eliminando ciertos componentes de alta frecuencia. La compresión de vídeo aprovecha estos tipos de redundancia alcanzando grandes tasas de compresión de datos mediante diferentes técnicas que serán brevemente descritas a lo largo de este capítulo.

3.2.1 Differential Pulse Code Modulation (DPCM). Estándar H.120

Uno de los primeros modelos que obtuvieron buenos resultados aplicados a la codificación de imagen fue DPCM, gracias a que este modelo es capaz de explotar la redundancia entre sucesivas muestras. Cada pixel es predicho a partir de una o más muestras transmitidas anteriormente. La versión más sencilla y simplificada utiliza simplemente el píxel anterior. En la Figura 3.1 se muestra un modelo más preciso que utiliza una media ponderada de los píxeles vecinos A, B y C.

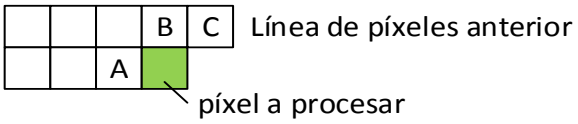


Figura 3.1. Píxeles vecinos en DPCM

El resultado de la resta entre el valor del pixel actual y el valor predicho, es conocido como error de predicción. Esta diferencia es cuantificada de forma que se reduce la precisión del sistema reduciendo el número de bits empleados. El resultado de la cuantificación es utilizada como entrada a un codificador entrópico. Este modelo puede ser aplicado para explotar tanto la redundancia espacial como la temporal. En el caso de la redundancia temporal, se emplearán para obtener la predicción, píxeles de otras imágenes pertenecientes a otros instantes de tiempo. Debido al proceso de cuantificación aplicado sobre la señal diferencial, se trata de una codificación con pérdidas, ya que no es posible obtener exactamente la imagen de entrada tras aplicar la descodificación. Es importante tener en cuenta que el codificador trabajará con píxeles reconstruidos para obtener la predicción, de esta forma obtendrá una predicción equivalente a la que se obtiene en el lado del descodificador que no dispone de los píxeles pertenecientes a la imagen original. El proceso de reconstrucción consiste en obtener un píxel a partir de los datos de entrada al codificador entrópico, siendo equivalente al proceso de descodificación tal y como se indica en la Figura 3.2. Por lo tanto, el codificador debe incluir para obtener los píxeles reconstruidos que utilizará en la predicción ciertos módulos comunes al descodificador.

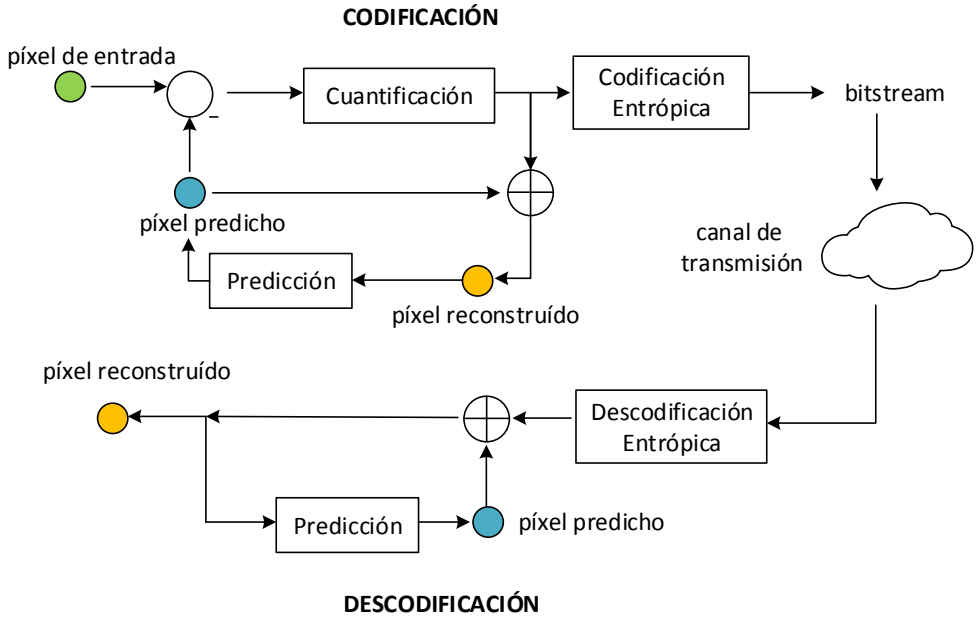


Figura 3.2. Esquema codificación/descodificación DPCM.

DPCM constituyó la base para crear el primer estándar de compresión de vídeo conocido como H.120 en 1984. La estandarización la realizó el *Comité Consultatif International Téléphonique et Télégraphique (CCIT)*, organismo que asentaría las bases para crear posteriormente la ITU. De las pruebas realizadas en numerosos estudios, se derivó que la redundancia temporal estaba siendo pobremente explotada y que para incrementar la tasa de compresión obtenida sin perder calidad visual se debería trabajar con conjuntos o bloques de píxeles como unidad básica de compresión en lugar de operar a nivel de píxel.

3.2.2 Transformada discreta de coseno. JPEG y H.261

Los trabajos posteriores realizados por ITU para crear un estándar de compresión de vídeo para videoconferencia estuvieron principalmente basados en el uso de transformada DCT. El principal objetivo del uso de esta transformada es trabajar en el dominio de la frecuencia de tal forma que se pueden distinguir claramente las distintas frecuencias que componen la señal de vídeo para posteriormente eliminar las componentes de alta frecuencia que no son percibidas por el HVS. La salida de este proceso proporciona un conjunto de coeficientes, que tras ser convenientemente ordenados, son cuantificados. La transformada no aporta ninguna compresión por sí misma, es la cuantificación posterior la encargada de eliminar los coeficientes irrelevantes y reducir el número de bits de los coeficientes que sí se codificarán.

Entre 1984 y 1988 el grupo *Joint Photographic Experts Group (JPEG)* tras varios estudios eligió la transformada DCT aplicada en bloques cuadrados de 8x8 píxeles como módulo fundamental de su codificador para imágenes estáticas. Esta decisión sirvió para consolidar su uso dentro de las investigaciones realizadas por ITU, que llevaron en 1989 a la definición del estándar H.261, cuyo principal ámbito de aplicación eran las videoconferencias utilizando una tasa de transmisión de datos de 64 kbit/s. La unidad básica de codificación de H.261 está compuesta por un bloque de 16x16 píxeles de luminancia y dos bloques de 8x8 píxeles para la crominancia (sólo soporta YCbCr 4:2:0).

Dicha unidad básica es conocida como macrobloque. Este estándar asentaría la arquitectura de codificación híbrida utilizada en los estándares de compresión de vídeo posteriores. Dicha arquitectura combina la predicción basada en compensación de movimiento, y la codificación basada en transformadas con cuantificación escalar, con una posterior codificación binaria entrópica. Antes de la codificación entrópica se realiza un reordenamiento de los coeficientes en función de la frecuencia, conocido como escaneo *zig-zag*. En la Figura 3.3, se muestra la arquitectura híbrida adoptada por el estándar H.261.

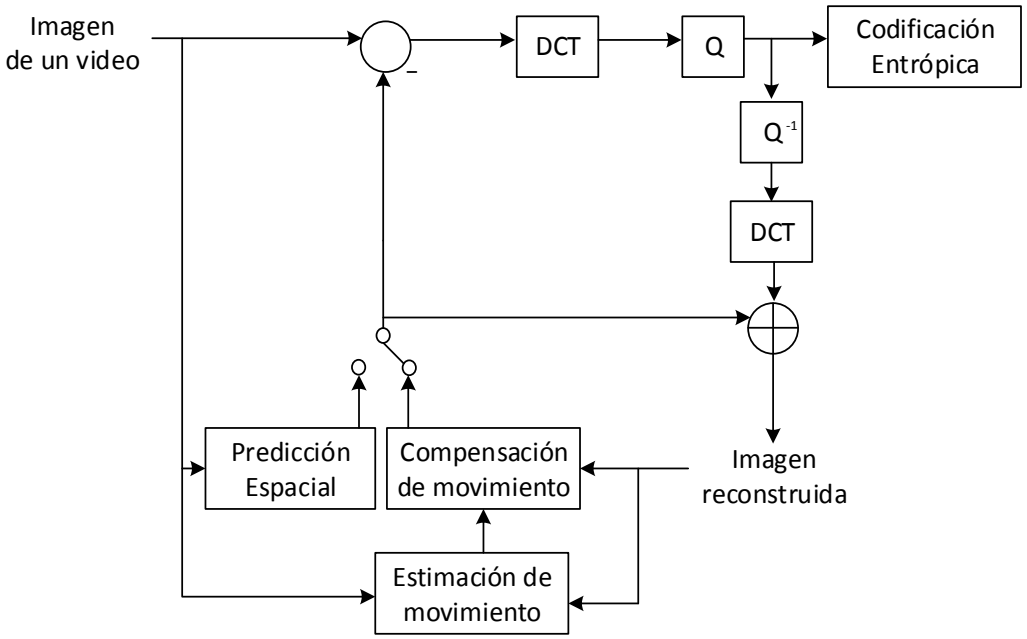


Figura 3.3. Arquitectura híbrida utilizada en el estándar H.261.

3.2.3 MPEG-1

A principio de la década de los 90, el grupo *Motion Picture Experts Group (MPEG)* comenzó a investigar técnicas de codificación para secuencias de vídeo muy activas con aplicación directa en el almacenamiento de películas en soportes digitales como el CD-ROM. Como fruto de este trabajo y empleando como base H.261, se definió el estándar de compresión de vídeo MPEG-1, alcanzando tasas de compresión más elevadas que su predecesor, estando optimizado para su uso utilizando tasas cercanas a 1.2 Mbit/s.

En este nuevo estándar se introduce el concepto de predicción temporal bidireccional, refiriéndose a la predicción que puede obtenerse de planos pasados, futuros o mediante una media de ambas predicciones. También se incorpora por primera vez el concepto de estimación de movimiento con precisión de ½ píxel. Básicamente, sobre la ventana de búsqueda utilizada para la estimación de movimiento se realiza una interpolación de forma que se pueden obtener vectores de movimiento de mayor precisión. Mientras que H.261 sólo soportaba las resoluciones CIF (352×288 píxeles) y QCIF (176×144), MPEG-1 permitía la utilización de multitud de resoluciones.

3.2.4 MPEG-2 (H.262)

Orientado al mercado *broadcast* y como evolución del estándar MPEG-1, el grupo MPEG publicó la primera versión del estándar ISO/IEC 13818-2 en 1996. Al mismo tiempo la organización ITU-T adoptó este mismo estándar para sistemas de comunicación bajo el nombre H.262. La principal novedad que introdujo respecto a su predecesor fue su aplicación para formatos de vídeo entrelazados. Su uso ha tenido gran impacto para la implantación de la televisión digital, el DVD y los enlaces por satélite. Gracias a la calidad visual que proporciona MPEG-2, se sustituyó el uso de las cintas VHS por el DVD para la visualización de películas en el mercado de consumo, aspecto en el que falló MPEG-1.

Junto a MPEG-2 apareció un protocolo de comunicación para audio, vídeo y datos conocido como *MPEG-2 Transport Stream* en el que mediante una multiplexación de los flujos de vídeo y audio se obtiene un único flujo en el que se insertan marcas de tiempo para asegurar la correcta sincronización de los dos flujos una vez sean descodificados en la parte receptora. Dentro de un mismo flujo de transporte pueden existir diferentes programas, cada uno de ellos con sincronizaciones audio/vídeo independientes, por lo que pueden coexistir diferentes programas de televisión.

3.2.5 MPEG-4 parte 2 (H.263)

La codificación MPEG-2 estaba orientada a un ancho de banda entre 4 y 9 Mbps, por lo que no era idóneo para aplicaciones que requerían el uso de canales con un ancho de banda reducido. Para este tipo de aplicaciones se trabajó en el estándar ISO/IEC 14496-2 (MPEG-4 parte 2) obteniendo importantes mejoras en cuanto a reducción en la tasa de datos generada respecto a su predecesor. Está parcialmente basado en ITU-T H.263 siendo un descodificador MPEG-4 capaz de descodificar H.263. H.263 es un estándar publicado en 1995 siendo una evolución del H.261 cuya principal aplicación era la videoconferencia.

Dentro de las características más relevantes respecto a sus predecesores, MPEG-4 destacó por el uso de la codificación orientada a objetos, diversas características que permiten la escalabilidad, interpolación de ¼ de píxel, posibilidad de uso de muestreo de color 4:4:4 y el manejo de escenas visuales sintéticas. Se introdujo la posibilidad de utilizar variaciones en el diseño del codificador/descodificador normativo para permitir el uso del estándar en multitud de aplicaciones. Dichas variaciones vienen definidas por 19 perfiles que describen las herramientas que pueden ser utilizadas en cada aplicación. Se trató de un proyecto muy ambicioso que buscaba abarcar todo tipo de aplicaciones, pero nunca llegaron a ser ampliamente utilizadas todas sus funcionalidades y fue paulatinamente sustituido por otros estándares con un enfoque más tradicional. La mayoría de aplicaciones que hicieron uso de MPEG-4 utilizaron el perfil básico para sustituir sistemas basados en MPEG-2, aunque no consiguió desbancar a MPEG-2 en la transmisión de señales de televisión.

3.2.6 H.264 (H.26L, MPEG-4 parte 10, AVC)

En 2003 surgió el estándar H.264 como proyecto de *Joint Video Team (JVT)*, una colaboración entre *ITU-T Video Coding Expert Group (VCEG)* e ISO/IEC JTC1 MPEG. También es conocido como MPEG-4 parte 10 o *Advanced Video Coding (AVC).* El principal objetivo de este proyecto era crear un estándar capaz de reducir a la mitad o menos la tasa de datos generada por sus predecesores para una misma calidad visual prefijada. Además, se quería dotar al estándar de una flexibilidad que le permitiese ser utilizado en una amplia variedad de aplicaciones utilizando resoluciones SD y HD, que fuese utilizado tanto para el mercado broadcast como para el almacenamiento en soporte físico, en redes de comunicaciones y en los sistemas multimedia de telefonía de ITU-T. En H.264 se abandona la codificación orientada en objetos de MPEG-4 y se retorna al enfoque tradicional de sus predecesores.

El estándar H.264 ha sido el precursor de la televisión digital de alta resolución, los discos Blu-ray y el estándar adoptado para la distribución de contenidos por Internet que permitió el éxito de por ejemplo, Youtube o Vimeo. Su uso en CCTV está actualmente consolidado, y existen numerosas plataformas hardware que permiten tanto su codificación como descodificación. Las GPU de Nvidia, AMD, ARM o Intel disponen de hardware embebido específico para su procesado lo cual ha permitido su amplia difusión en la electrónica de consumo.

Dentro de las principales mejoras que introdujo este estándar, está el uso del *Deblocking Filter* en el bucle de reconstrucción, la utilización de la transformada 4x4 entera y la implementación del *Context Adaptative Binary Arithmetic Coder (CABAC).* El filtro de deblocking permite la reducción de la percepción de bloques cuadrados debidos intrínsecamente a la compresión (Figura 3.4). El uso de la transformada entera permite que la imagen reconstruida por el descodificador sea una imagen totalmente igual a la obtenida en el proceso de reconstrucción del codificador, lo que mejora la predicción y simplifica considerablemente la depuración de errores en el sistema. Además, al estar implementada con operaciones de sumas, restas y desplazamientos y usar una aritmética de 16 bits, puede implementarse fácilmente de una manera óptima. El CABAC permite reducir hasta en un 10% la tasa de datos generada respecto a su predecesor *Context-Adaptative Variable Length Code (CAVLC).* En determinados perfiles de H.264 se permite el uso de CABAC o CAVLC, aunque en el perfil *Baseline* únicamente se puede utilizar CAVLC.

Figura 3.4. Artefactos visuales intrínsecos a la codificación basada en bloque (efecto bloque).

3.2.7 H.265 o HEVC

Continuando con la colaboración establecida en el estándar H.264 entre ITU-T VCEG y ISO/IEC MPEG surgió *Joint Collaborative Team on Vídeo Coding (JCT-VC),* grupo orientado a desarrollar el estándar H.265 también conocido como *High Efficiency Vídeo Coding (HEVC).* El objetivo que se estableció coincide con el de anteriores colaboraciones, obtener una reducción en la tasa de datos generada del 50% respecto a su predecesor. La primera versión de este estándar se publicó en 2013, y su implantación ha permitido la transmisión de vídeo en resolución 4K. Aporta considerables mejoras respecto al H.264 en cuanto al CABAC, el filtro de deblocking y la predicción tanto espacial como temporal. En este estándar se introduce por primera vez una estructura muy flexible de división de la imagen que permite el uso de múltiples tamaños en las unidades básicas de codificación, predicción y transformación. HEVC aumenta el tamaño de la unidad básica de codificación de 16x16 que utilizaban sus predecesores hasta 64x64, permitiendo también el uso de 32x32, lo cual ha demostrado que para determinadas secuencias resulta muy beneficioso desde el punto de vista de la reducción de tasa de datos.

En el apartado 3.3 se explican en detalle las diferentes técnicas que componen H.265, lo que permitirá definir una serie de conceptos necesarios para la comprensión de los diferentes algoritmos de optimización/simplificación que se han desarrollado en la presente Tesis.

3.2.8 Software de referencia/modelo de test

De manera conjunta a cada estándar es desarrollado un software o modelo de referencia para implementar la funcionalidad definida. Dicho software es liberado junto a un documento que describe su funcionamiento denominado modelo de test. El software de referencia y modelo de test son continuamente revisados y actualizados con los refinamientos que se realizan en las reuniones posteriores a la liberación de la primera versión de estándar.

El software de referencia es normalmente utilizado para la evaluación de las diferentes técnicas que componen un estándar. Además, sirve como soporte al estándar para facilitar la comprensión de los algoritmos que lo constituyen y permite establecer una cota máxima de la tasa de compresión que se puede alcanzar para una determinada calidad visual. Los nuevos algoritmos, optimizaciones y/o simplificaciones que aparecen en la literatura relacionada con la compresión de vídeo utilizan el software de referencia para la evaluación de los resultados. Los resultados se presentan como una comparativa respecto al software de referencia sin modificar, en términos de tasa de datos generada, calidad visual obtenida y reducción de tiempos de ejecución. De esta forma, los resultados de los distintos trabajos publicados pueden fácilmente compararse entre sí. La implementación del software de referencia no se encuentra optimizada, puesto que el objetivo que persigue no es su ejecución en tiempo real.

3.3 Conceptos Compresión de Vídeo

En apartados anteriores se ha definido la compresión de vídeo, su funcionalidad y se ha resumido su evolución en los últimos años a través de su normalización por medio de diferentes estándares. En este apartado, se procederá a definir brevemente aspectos fundamentales de la compresión de vídeo que servirán como base de conocimiento para comprender las nuevas técnicas que componen el estándar HEVC y las optimizaciones que se han realizado sobre ellas.

3.3.1 Codificación basada en bloques

La compresión de vídeo está basada en la explotación de la redundancia temporal (predicción inter), espacial (predicción intra) y en el espacio de la frecuencia (aplicando transformadas discreta enteras de seno y de coseno), de forma que se elimina toda aquella información que no sea estrictamente necesaria para representar el vídeo original usado como entrada al codificador. Como se comentó previamente, tras la masiva utilización de DPCM y posteriores estudios y pruebas, la codificación a nivel de píxel ofrecía una escasa explotación de las redundancias existentes por lo que se comenzó a trabajar utilizando conjuntos de píxeles o bloques.

En el dominio temporal, en las secuencias de vídeo suele existir una gran correlación entre las distintas imágenes que las componen. La explotación de la redundancia temporal consiste en describir la imagen que se está codificando en función del movimiento que existe respecto a otras imágenes dentro de la secuencia de vídeo. La redundancia espacial hace uso únicamente de la información disponible dentro de la imagen que se pretende

codificar, explotando las similitudes entre píxeles vecinos. Finalmente, la redundancia en el espacio de la frecuencia se consigue mediante el uso de transformadas que permiten trabajar en el dominio de la frecuencia y que permiten eliminar toda la información redundante en dicho espacio mediante un proceso de cuantificación sobre los coeficientes generados.

Toda compresión lleva asociada una pérdida de información visual conocida como distorsión que debe de ser minimizada. Además de la explotación de los diferentes tipos de redundancia, la compresión de vídeo hace uso de un esquema de codificación entrópica que reduce el número de datos generados sin introducir distorsión.

La codificación basada en bloques presenta otras ventajas, como ser viable computacionalmente, es completamente compatible con el tamaño rectangular de las imágenes y se adapta a los algoritmos utilizados en las transformadas que trabajan a nivel de bloque. Su principal desventaja es que no se adapta adecuadamente a la forma de los objetos, por lo que el uso de tamaño de bloque variable para adaptarse a las diferentes estructuras que componen una imagen puede aportar una importante mejora en la eficiencia de codificación y por ello, HEVC ha adoptado una estructura de división de la imagen recursiva y que considera múltiples tamaños de bloque.

3.3.2 Predicción

Es un proceso realizado durante la codificación cuyo principal objetivo es reducir la cantidad de datos que son generados por medio de la explotación de la redundancia espacial y temporal. A grandes rasgos, este proceso consiste en obtener un residuo tras realizar la resta entre el bloque que se pretende codificar y una predicción de dicho bloque.

Si la predicción se obtiene haciendo uso únicamente de información contenida en el mismo plano que se está codificando (redundancia espacial) se denomina predicción intra. En el caso de que la predicción se obtenga a partir de planos previamente codificados (redundancia temporal) adquiere el nombre de predicción inter.

Cuanto más preciso sea el residuo, menos energía contendrá y menos datos serán generados en la codificación.

3.3.2.1 Predicción Intra

El objetivo de la predicción intra es la obtención de un residuo tras la resta del bloque que se pretende codificar y una predicción espacial obtenida a partir de píxeles de bloques vecinos previamente codificados. El set de píxeles vecinos disponibles está constituido por un conjunto distinto en función del estándar de vídeo utilizado. En la Figura 3.5 se indican los píxeles vecinos que pueden ser utilizados en H.264 para un bloque de tamaño 4x4 y la nomenclatura asociada. El conjunto está compuesto por píxeles pertenecientes al bloque inmediatamente superior (píxeles A, B, C, D), al izquierdo (I, J, K, L), superior izquierdo (M) y superior derecho (E, F, G, H). Las muestras obtenidas tras la predicción son las indicadas con las letras a-p en la Figura 3.5.

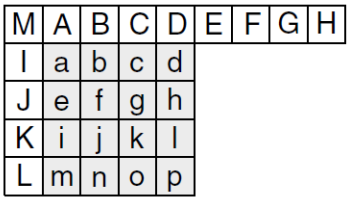


Figura 3.5. Muestras obtenidas tras la predicción (a-p) y píxeles vecinos para un bloque 4x4 en H.264.

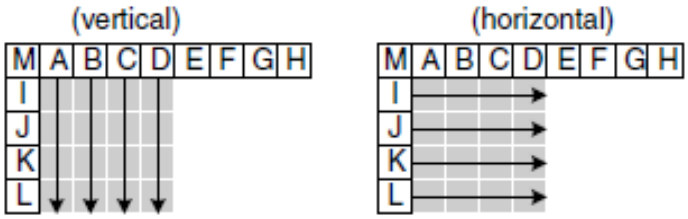


Figura 3.6. Modos de predicción vertical y horizontal en H.264.

Los modos de predicción intra determinan cómo se obtiene la predicción del bloque a codificar a partir de los píxeles vecinos previamente reconstruidos. En la Figura 3.6 se indican los píxeles vecinos que son utilizados para los modos de predicción vertical y horizontal. Como se puede observar, para el modo vertical, los píxeles a, e, i y m son obtenidos a partir del píxel A; los píxeles b, f, j, n a partir del píxel vecino B y así sucesivamente. En el modo horizontal, los píxeles a, b, c y d son obtenidos a partir del píxel I del bloque 4x4 vecino izquierdo.

Las componentes de luminancia y crominancia son tratadas independientemente. Los planos de vídeo en los que únicamente se utiliza predicción espacial se conocen como planos tipo I (*I-frames*).

3.3.2.2 Predicción Inter

La predicción inter reduce la redundancia entre los planos codificados mediante la codificación de un residuo formado por la resta de la imagen de entrada menos una predicción temporal. La predicción temporal se obtiene por medio del análisis del movimiento del plano que se pretende codificar respecto a planos codificados en instantes anteriores. Estos planos utilizados como referencias, pueden ser de instantes anteriores o futuros en el orden de presentación. Los planos reconstruidos que son utilizados como referencias son almacenados en las listas de referencias.

Para hacer posible el uso de referencias futuras, el orden de presentación y el orden de codificación/descodificación que presentan los planos involucrados es distinto. En la Figura 3.7, se muestra el orden que siguen los planos en la entrada al codificador, siendo éste el orden de presentación en el que serán mostradas las diferentes imágenes en un monitor para la reproducción del vídeo tras la descodificación.

Los planos donde se utilizan únicamente referencias de instantes anteriores en el orden de presentación son conocidos como planos tipo P (*P-frames*). Si en el plano actual se utilizan para su codificación tanto referencias pasadas como futuras en orden de presentación, los planos son conocidos como planos tipo B (*B-frames*). Para poder hacer uso de referencias futuras se requiere introducir un retardo inicial en la codificación de forma que cuando comience la codificación de un plano tipo B ya estén codificadas y almacenadas en memoria las referencias que necesita. Este retardo inicial es igual al

número de planos tipo B que se estén utilizando. En la Figura 3.7, para poder codificar el plano B1 se requiere haber codificado previamente los planos I0 (referencia pasada) y P3 (referencia futura). El proceso equivalente en el descodificador es mostrado en la Figura 3.8. Como se pude apreciar el orden de descodificación es igual que el orden de codificación, y el orden de presentación es exactamente el mismo que el orden de entrada al codificador.

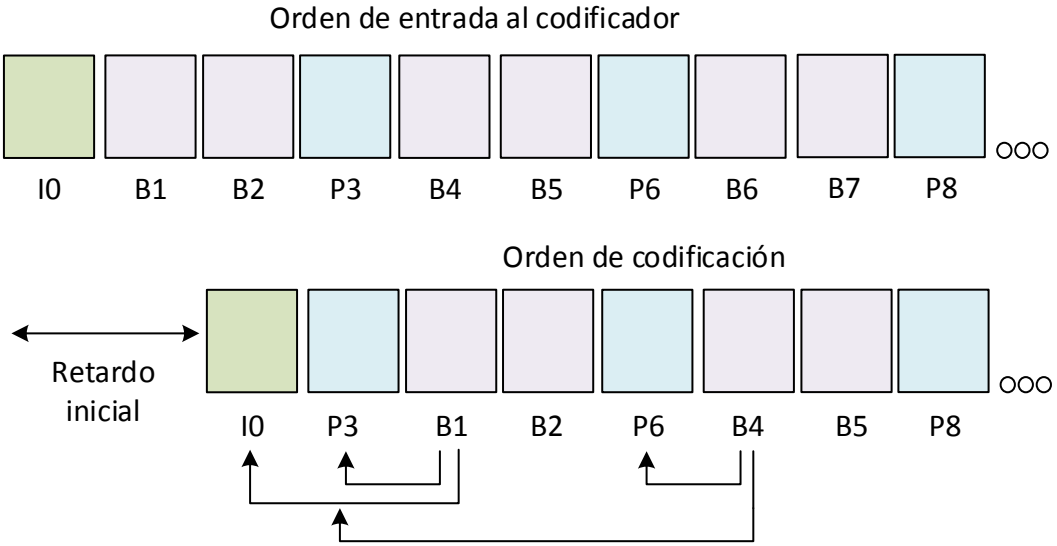


Figura 3.7. Orden de presentación y de codificación para una secuencia de vídeo utilizando dos planos tipo B.

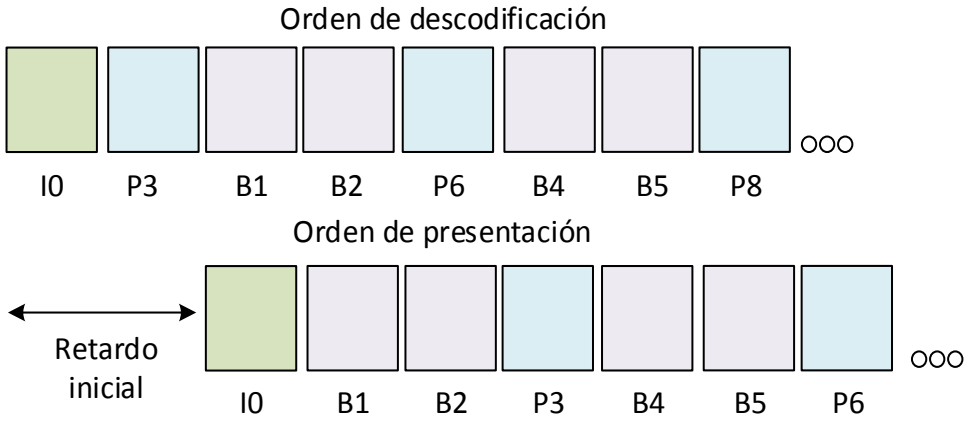


Figura 3.8. Orden de presentación y de descodificación para una secuencia de vídeo utilizando dos planos tipo B.

3.3.2.2.1 Estimación de movimiento

El proceso mediante el cual se realiza una búsqueda en los planos de referencia de un bloque que coincida o se asemeje considerablemente al bloque que se pretende codificar, se conoce como estimación de movimiento. El resultado de este proceso son las coordenadas en los ejes X e Y del bloque encontrado en la imagen de referencia. El proceso

de búsqueda no está estandarizado, por lo que el algoritmo de búsqueda se presta a numerosos diseños y optimizaciones.

Uno de los algoritmos tradicionalmente utilizados y más ampliamente difundidos consiste en realizar la comparación entre el bloque de imagen a codificar de tamaño WxH píxeles con diferentes bloques de WxH píxeles en distintas posiciones de las imágenes utilizadas como referencia. Un criterio de evaluación muy extendido es la resta entre el bloque candidato y el bloque a codificar, considerando que el mejor candidato será aquel que minimiza la energía del residuo obtenido. A su vez qué imágenes utilizar como referencia, el número de posiciones a evaluar y las coordenadas de dichas posiciones, son parámetros que determinan la eficacia y eficiencia del algoritmo de búsqueda.

La zona donde se realiza la búsqueda se conoce como área de búsqueda. Es habitual, que dicho área se centre en coordenadas extraídas de la información de movimiento correspondiente a los vecinos espaciales y temporales del bloque que se pretende codificar, considerando que existe una alta probabilidad de que el movimiento del bloque sea muy similar al de sus bloques vecinos. También se incluyen las coordenadas del bloque a codificar, para considerar la posibilidad de que no exista movimiento o sea muy pequeño. Una vez establecida la posición en la que se centrará la zona de búsqueda es importante establecer su tamaño, cuanto mayor sea su tamaño mayor será la capacidad del algoritmo de localizar objetos que se hayan desplazado considerablemente entre los distintos planos que componen la secuencia de vídeo. En la Figura 3.9 se muestra un ejemplo de área de búsqueda, en el que el bloque a codificar ha sido encontrado en una referencia en las coordenadas (x1, y-1). El vector que representa el movimiento del bloque encontrado respecto a la posición de inicio de la búsqueda se conoce como vector de movimiento.

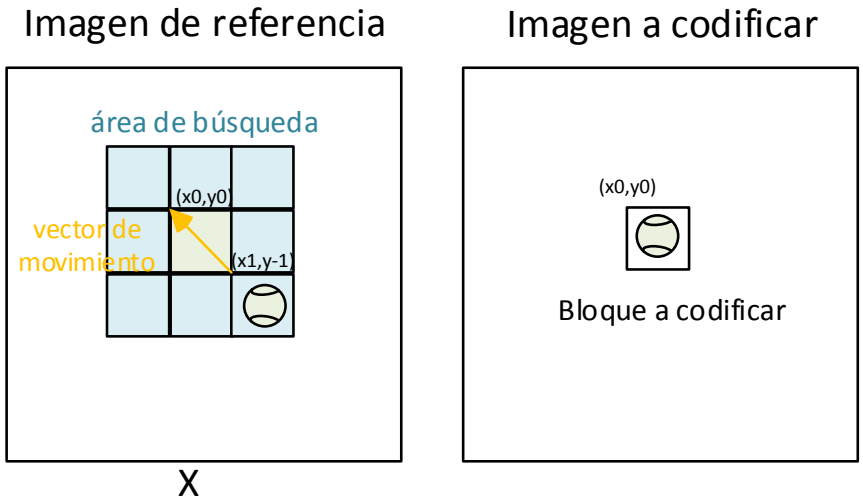


Figura 3.9. Ejemplo de estimación de movimiento.

Dentro del área de búsqueda se evaluarán distintos puntos conforme a un patrón de búsqueda. En la Figura 3.10, se muestra un patrón de búsqueda en espiral que evalúa todos los puntos de la zona de búsqueda de una forma exhaustiva (*full-search*) para un bloque de tamaño 4x4. En cada punto de la espiral indicado por el final de una flecha, se realizaría una evaluación, restando el bloque candidato de tamaño 4x4 de la imagen de referencia con el bloque 4x4 a codificar.

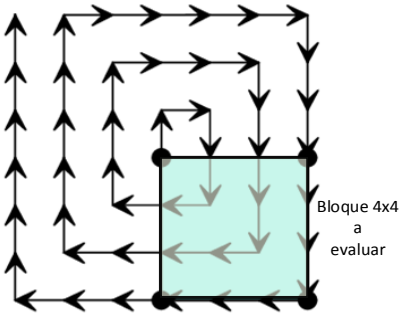


Figura 3.10. Búsqueda exhaustiva para tamaños de bloque 4x4.

En la Figura 3.11, se indican distintos patrones de búsqueda utilizados habitualmente como alternativa a la búsqueda exhaustiva. Los puntos en rojo muestran el punto de partida de la búsqueda, que pueden ser distintas posiciones dentro de la zona de búsqueda. Los puntos que pertenecen a un mismo color son los evaluados en cada iteración.

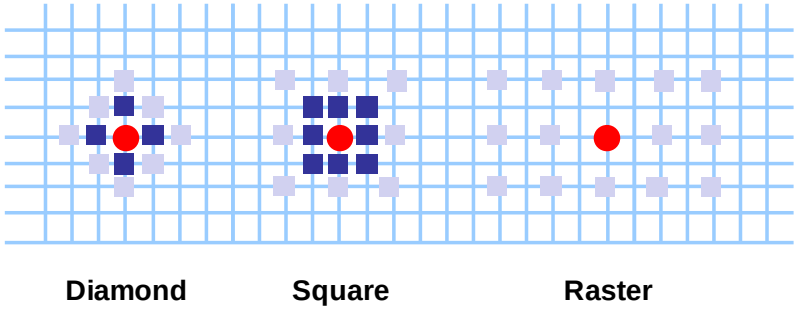


Figura 3.11. Patrones de búsqueda habituales.
Imagen extraída de [2].

3.3.2.2.2 Compensación de movimiento

La compensación de movimiento se basa en la obtención de una predicción a partir de los vectores de movimiento resultado de la estimación de movimiento. Dicha predicción está formada por los píxeles que conforman el bloque situado en la imagen de referencia en las coordenadas indicadas por los vectores de movimiento. El bloque obtenido como resultado de la compensación de movimiento se resta al bloque original de la imagen entrada obteniendo el residuo que es transformado, cuantificado y posteriormente codificado y transmitido o almacenado junto a información del movimiento que describe su posición. Dicha información incluye los vectores de movimiento que indican en las coordenadas X e Y, el desplazamiento del bloque respecto a una posición conocida, y la información respecto a la imagen que se utilizó como referencia. Por lo tanto, el vector de movimiento que se codifica realmente es la diferencia (*Motion Vector Difference, MVD*) entre las coordenadas del bloque resultado de la búsqueda que se asemeja al bloque a codificar (*motion vector, MV*) y unas coordenadas prefijadas conocidas como vector de movimiento predicho (*Motion Vector Predictor, MVP*).

$$MVDx = MVx - MVPx,$$
$$MVDy = MVy - MVPy. \quad (3.1)$$

El proceso de derivación de MVP depende del estándar de codificación empleado pero coincide en que siempre es obtenido a partir de vecinos espaciales y/o temporales. El índice de referencia indica dentro de una lista que contiene las imágenes de referencia que se pueden utilizar, cuál fue utilizada para codificar el bloque de imagen actual.

3.3.2.2.3 Interpolación

En determinadas secuencias de vídeo puede obtenerse una mejor predicción que reduzca la energía del residuo por medio de una interpolación realizada sobre los píxeles de la imagen de referencia donde se realiza la búsqueda de movimiento. Si de cada dos píxeles se obtiene un píxel interpolado, la interpolación se conoce como interpolación de medio píxel (*sub-pixel* o *half-pel, hpel*). La estimación de movimiento basada en medio píxel evalúa tanto los píxeles sin interpolar como los obtenidos de la interpolación, aportando este refinamiento en la búsqueda una importante mejora en la precisión a la hora de detectar el movimiento de un objeto. Existe la posibilidad de realizar una nueva iteración en el proceso de interpolación utilizando píxeles sin interpolar e interpolados a nivel de medio píxel, obteniendo píxeles interpolados a nivel de cuarto de píxel (*quarter-pixel* o *quarter-pel, qpel*). Con cada nuevo refinamiento se mejora la predicción obtenida a expensas de un incremento en los tiempos de cómputo, pero la mejora va disminuyendo a medida que se van realizando nuevas interpolaciones. La mejora que aporta medio píxel es bastante grande, cuarto de píxel mejora moderadamente el resultado tras aplicar medio píxel, y si se continúa interpolando a un octavo de píxel se obtendría una pequeña mejora respecto al uso de cuarto de píxel.

En la Figura 3.12 se representa una búsqueda de movimiento que incorpora una interpolación de cuarto de píxel. Tras obtener el mejor candidato a nivel de píxel sin interpolar, se realiza una interpolación y una nueva búsqueda a nivel de medio píxel. A partir del mejor punto obtenido en la búsqueda a nivel de medio píxel, se realiza una interpolación a nivel de cuarto de píxel y se realiza una nueva búsqueda. El mejor punto obtenido en esta última búsqueda, se utiliza para finalmente obtener la predicción que será utilizada en la compensación de movimiento. La precisión en la interpolación que se puede utilizar en la compensación de movimiento viene definida por el estándar que se utilice, H.264 por ejemplo, permite una precisión de hasta cuarto de píxel, mientras que HEVC alcanza hasta un octavo.

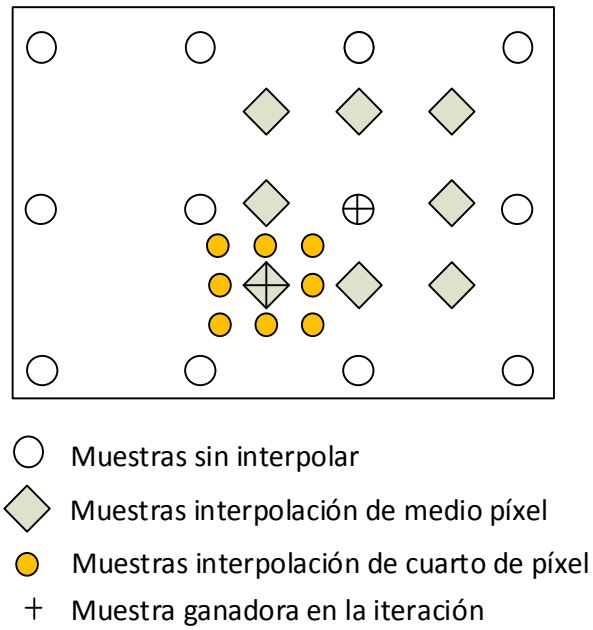


Figura 3.12. Estimación de movimiento a nivel de cuarto de píxel.

3.3.3 Transformación

Mediante este proceso el residuo es convertido al dominio de la frecuencia para su posterior tratamiento. En los estándares de compresión de vídeo se han escogido diferentes tipos de transformadas, pero todas ellas presentan unas características comunes. Trabajan a nivel de bloque utilizando diferentes componentes independientes, utilizan operaciones computacionalmente abordables, y para posibilitar la descodificación, son procesos reversibles. La transformada discreta de seno (*Discrete Sine Transform, DST*) y la DCT son ejemplos de este tipo de transformadas. El principal problema que presenta este tipo de transformadas es que provocan artefactos visuales en las fronteras entre los distintos bloques. En tratamiento de imágenes se utilizan otro tipo de transformadas que operan a nivel de imagen completa, como la transformada discreta de Wavelet *(Discrete Wavelet Transform, DWT)* utilizada en JPEG. Este tipo de transformada tiene unos requerimientos de uso de memoria muy superiores a las transformadas basadas en bloques, y no son adecuadas para ser utilizadas en sistemas que utilicen una compensación de movimiento ejecutada a nivel de bloque.

La transformación en sí no produce ningún tipo de compresión, puesto que de cada residuo se obtiene un coeficiente. Además, dicho coeficiente requiere para su representación de un número de bits superior al necesario para representar un píxel. La verdadera utilidad de la transformada reside en que valores de coeficientes próximos a cero pueden ser eliminados sin afectar apreciablemente a la reconstrucción de una imagen tras aplicar la transformación inversa. Por otro lado, el sistema visual humano es menos sensible a coeficientes de alta frecuencia, por lo que estos coeficientes pueden ser eliminados sin afectar a la percepción visual. La eliminación de estos coeficientes antes de aplicar la codificación entrópica supone una importante reducción en la tasa de datos generada.

3.3.4 Cuantificación

La cuantificación permite reducir el número de bits con los que se representa un coeficiente, por lo que reduce su precisión, así que el rango de valores que puede representar dicho coeficiente es menor. El grado en el cual dichos bits es reducido depende del valor del parámetro de cuantificación (*Quantization Parameter, QP*). Cuanto mayor sea QP, mayor número de bits serán eliminados. Tras la cuantificación se aumentará el número de coeficientes cercanos a cero de los que se puede prescindir. Varios estándares permiten el uso de matrices de cuantificación, mediante las cuales se aplican distintos escalones de cuantificación en función de la posición del coeficiente dentro del bloque transformado. Las posiciones de los coeficientes dentro del bloque transformado indican la frecuencia a la que pertenecen, por lo que las matrices de cuantificación permiten aplicar distintos escalones de cuantificación en función de la frecuencia.

3.3.5 Reordenamiento de coeficientes (scan)

Considerando la transformada DCT, al ser la utilizada en los últimos estándares de compresión H.264 y HEVC, se puede observar que tras aplicar el proceso de cuantificación se obtiene una gran cantidad de coeficientes con valor cero o cercano a cero. También se observa que los coeficientes con valores significativos se concentran en las inmediaciones al coeficiente con coordenadas (0,0) dentro del bloque transformado. Dichas posiciones se corresponden con los coeficientes de baja frecuencia, por lo que tras aplicar la cuantificación se han eliminado gran parte de los coeficientes de alta/media frecuencia. El coeficiente con coordenadas (0,0) se le denomina coeficiente DC y al resto, coeficientes AC. El coeficiente DC, tras la transformación inversa representa la media de los valores de los píxeles del bloque.

El reordenamiento de coeficientes, también conocido como *zig-zag scan* permite agrupar los coeficientes distintos de cero para representar el bloque de una manera más compacta para su posterior tratamiento. En la Figura 3.13 se muestra el reordenamiento que se realiza en el estándar H.264 para la codificación de formatos de vídeo progresivo.

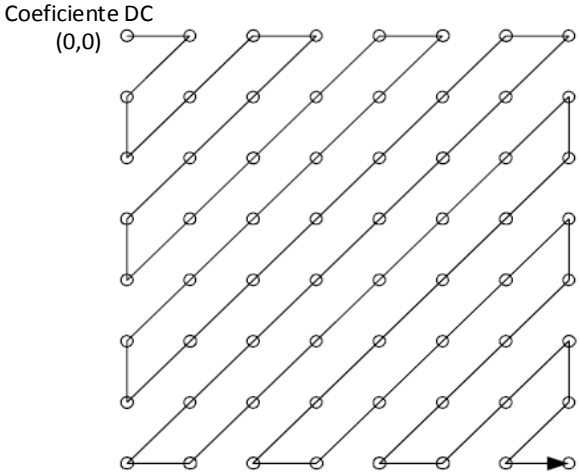


Figura 3.13. H.264 zig-zag scan.

3.3.6 Reconstrucción

Para obtener las imágenes de referencia, el codificador de vídeo incluye dentro de su estructura ciertos módulos comunes a un descodificador, en concreto la parte referente a la reconstrucción de las imágenes. Para realizar la reconstrucción, al residuo de entrada al codificador entrópico se le aplica una transformada y una cuantificación inversa. Al resultado de estos procesos se le suma la predicción, obteniendo un bloque reconstruido de la imagen. Las imágenes reconstruidas son almacenadas en un búfer para ser utilizadas como referencia en futuras estimaciones de movimiento. Por lo tanto, este búfer contiene un conjunto de imágenes equivalente al que contiene el descodificador al otro lado del canal de comunicación tal y como se indica en el esquema mostrado en la Figura 3.14. En el caso de la predicción intra, se utilizan dentro de la imagen que se está codificando, los píxeles reconstruidos de bloques vecinos espaciales para obtener el mejor modo de predicción intra.

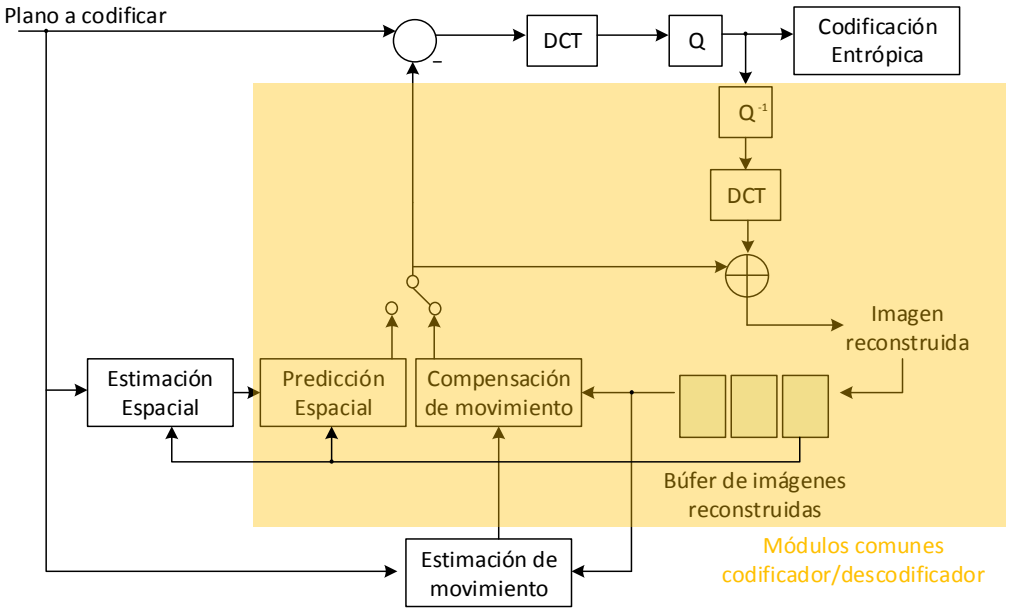


Figura 3.14. Módulos comunes codificación/descodificación.

3.3.7 Codificador entrópico

El codificador entrópico es el encargado de convertir una serie de símbolos que representan la información que define la secuencia de vídeo, conocidos como elementos de sintaxis (*syntax elements*), en una cadena de bits reducida que sea adecuada para su transporte a través de un canal de transmisión o para su almacenamiento. Este proceso debe de ser completamente reversible de forma que el valor de los símbolos codificados pueda ser recuperado en el descodificador. A su vez, un codificador entrópico debe de aportar una tasa de compresión elevada para reducir las necesidades de ancho de banda para la transmisión o de capacidad de almacenamiento. Para conseguir elevar la tasa de compresión, los codificadores entrópicos hacen uso de una codificación predictiva. Esta codificación hace uso de la gran correlación que existe entre distintos símbolos, como pueden ser los valores en una coordenada (X o Y) de los vectores de movimiento de bloques

adyacentes; o los valores de los coeficientes DC. De esta forma, se codifica la información referente a la diferencia entre el valor predicho obtenido de los vecinos y el valor real.

El uso de codificación de tamaño variable (*variable-length coding, VLC*) permite reducir aún más la tasa de datos generada por medio de la conversión de los símbolos a códigos (*codewords*) de longitud variable. En este tipo de codificación se asignan los códigos de menor número de bits a los símbolos con mayor probabilidad de ocurrencia y los de mayor número de bits a los de menor probabilidad. Los primeros esquemas de codificación de este tipo, asignaban un número entero de bits a cada símbolo (como el caso de la codificación Huffman), lo cual presenta la clara desventaja de que el número de bits teóricamente óptimo debe depender de la información que contenga el símbolo, pudiendo ser un número fraccionario. Los codificadores aritméticos resuelven este problema convirtiendo una secuencia de símbolos en un único número fraccionario. A cada símbolo se le asigna un sub-rango de probabilidad dentro del rango total de 0.0 a 1.0 dependiendo de la probabilidad de ocurrencia. Cada vez que un símbolo es codificado, el rango se progresivamente reducido. Al final de la codificación, la secuencia de símbolos puede ser representada transmitiendo un número fraccionario que esté dentro del rango final obtenido tras este proceso de codificación recursivo. De esta forma se puede obtener el número de bits óptimo requerido para cada símbolo. Por otro lado, el éxito de la codificación entrópica depende directamente de la utilización de modelos precisos de la probabilidad de ocurrencia de cada símbolo. La codificación aritmética basada en contexto (*Context-based Arithmetic Coding, CAE*), utiliza las características espaciales y temporales de los símbolos para realizar la estimación de probabilidad de cada símbolo. En la codificación aritmética binaria, los únicos posibles valores de símbolo son ‘0’ y ‘1’.

3.3.8 Control de Tasa

En la codificación basada en bloques, si se utiliza un valor de QP constante para todo el plano, cada unidad básica de codificación generará un número distinto de bits dependiendo del contenido de la imagen. Por lo tanto, el número de bits generado para cada imagen variará dependiendo del contenido, de forma que sea imposible controlar el número de bits obtenidos por el codificador en un determinado espacio de tiempo.

La variación en la tasa de datos generada es problemática en aplicaciones que usen un canal de transmisión a tasa constante. Por ejemplo, las redes IP pueden soportar variaciones en la tasa de transmisión siempre y cuando la media permanezca constante, ya que debe de ajustarse a posibles problemas de congestión. Por otro lado, el ancho de banda de un canal de transmisión es limitado, con un valor de QP que permanezca invariable es imposible asegurar que la tasa de datos que se genera sea constante, pudiendo incluso sobrepasar la tasa de datos soportada por el canal. La capacidad de almacenamiento de un dispositivo es otro factor a tener en cuenta, ya que si se desea grabar un vídeo se debe de tener en cuenta la capacidad de almacenamiento del dispositivo físico para no sobrepasarla.

El control de tasa (*rate control, RC*) es un algoritmo que dinámicamente ajusta el valor de QP para alcanzar una tasa de datos objetivo. Para conseguir esta tasa, el algoritmo calcula un número de bits objetivo para un determinado grupo de imágenes, una determinada partición de la imagen y/o un conjunto de unidades básicas de codificación. El número de bits objetivo es continuamente recalculado en función del número de bits que se ha generado en estadios previos de la codificación. El número de bits obtenido durante la codificación es reducido aumentando el valor del QP, lo que reduce la calidad visual obtenida. Por el contrario, cuando el presupuesto de bits lo permite, se aumenta el número

de bits generado en la codificación reduciendo el valor de QP y consiguiendo por lo tanto, aumentar la calidad visual.

El algoritmo de control de tasa no está especificado dentro de ningún estándar, pero es una parte fundamental del proceso de codificación.

3.3.9 Optimización distorsión/tasa (Rate-distortion optimization, RDO)

Como se ha comentado previamente, en la predicción tanto espacial como temporal, existen diversos bloques candidatos. El codificador debe ser capaz de evaluar las diferentes posibilidades para encontrar la mejor predicción, siendo ésta la que proporcione la mayor calidad visual y el menor número de bits posible. Por lo tanto, hay que adoptar una solución de compromiso, ya que existirán soluciones en el que una mínima mejora de calidad suponga un excesivo incremento del número de bits generados. La optimización distorsión/tasa, es un método para resolver este problema minimizando la pérdida de calidad visual (distorsión) frente al número de bits requeridos (tasa). La decisión óptima es la que permita obtener el residuo con menor energía, lo que supone que la predicción presenta un elevado grado de similitud respecto a la imagen original que se pretende codificar.

La desviación de la predicción respecto al bloque original es medida utilizando métricas como *Mean Square Error (MSE)*, *Mean Absolute Error (MAE)* o *Sum of Absolute Errors (SAE)* también conocida como *Sum of Absolute Differences (SAD)*. Las Ecuaciones 3.2 describen estas medidas.

$$MSE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(C_{ij} - R_{ij})^2, \quad (3.2)$$
$$MAE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|C_{ij} - R_{ij}|^2,$$
$$SAE = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|C_{ij} - R_{ij}|.$$

Donde C_{ij} representan los píxeles del bloque original a codificar y R_{ij}, los píxeles del bloque candidato reconstruido que se está evaluando de dimensiones NxN.

La métrica SAD es la más utilizada, debido a que es la métrica computacionalmente menos costosa, aunque en numerosas implementaciones ha sido sustituida por *Sum of Absolute Transformed Differences (SATD)*, que utiliza en lugar del residuo, el resultado de aplicar al residuo la transformada Hadamard. SATD incrementa el gasto computacional asociado, pero obtiene una mejor precisión en la medida de energía.

La otra restricción a tener en cuenta es la tasa de datos generada, existiendo un valor máximo y un mínimo, impuestos por el algoritmo de control de tasa. Cada decisión tomada por el codificador, conlleva un conjunto de símbolos que deben codificarse para su posterior envío. La codificación de cada símbolo o conjunto de símbolos relativos a un mismo dato, supone un determinado número de bits conocido como coste. La estimación de dichos bits es multiplicado por un valor lagrangiano (λ) que representa la relación entre el número de bits y un determinado nivel de calidad visual.

Por lo tanto, el problema asociado a RDO puede formularse como se indica en la Ecuación 3.3. Se trata de encontrar para una determinada decisión tomada en el codificador, el valor de J que minimice el problema.

$$Min\ \{J\},$$
$$J = D + \lambda * R. \qquad (3.3)$$

Siendo D la medida de distorsión. R representa el número de bits que son requeridos para señalizar una determinada decisión tomada en el codificador y λ, el valor lagrangiano que relaciona R con un determinado nivel de calidad visual.

Las diferentes decisiones a las que se enfrenta el codificador son los distintos vectores de movimiento disponibles, modos de predicción intra, índices de referencia, tamaño del bloque a codificar, etc. El valor de λ suele variar dependiendo de la decisión que se tome, y no se trata de un valor definido en el estándar ya que corresponde a la parte codificadora. En la distribución asociada al codificador en el software de referencia de cada estándar se definen una serie de valores de lagrangianos que suelen ser utilizados en las aplicaciones comerciales. La obtención del coste en bits asociado a una determinada predicción (R) requiere la codificación completa del bloque, por lo que se debe obtener el residuo, realizar su posterior transformación, cuantificación y aplicar a los coeficientes obtenidos la codificación entrópica. Debido a ello, el proceso RDO es computacionalmente muy costoso y su optimización y/o simplificación es objeto de numerosos estudios. Es habitual que en algunos procesos como la estimación de movimiento a nivel de sub-píxel se elimine el uso de RDO y se utilice para la comparación entre los diversos candidatos una función de coste asociado que sólo contemple la distorsión como SAD o SATD.

3.4 Referencias

[1] Richardson, I. E. (2002). Video Codec Design: Developing Image and Video Compression Systems. Chichester, UK: Wiley.

[2] McCann K., Rosewarne C., Bross B., Naccari M., Sharman K., Sullivan G. (2014). High efficiency video coding (HEVC) encoder description 0v16 (HM16). JCT-VC High Efficiency Video Coding N14 703.

Capítulo 4

Introducción al estándar HEVC

4.1 Arquitectura

HEVC está basado en la ampliamente conocida arquitectura de codificación híbrida basada en bloques, la cual combina la predicción basada en compensación de movimiento y la codificación basada en transformadas con una posterior codificación entrópica binaria. Dicha arquitectura ha sido utilizada por otros estándares de codificación como su predecesor el H.264, pero a diferencia de este último, HEVC emplea una estructura de árbol para la subdivisión de la imagen en bloques conocida como *quad-tree coding block partitioning*. Dicha estructura permite la utilización de tamaños de bloque mayores y múltiples combinaciones para adaptarse a las distintas características de la imagen. El diagrama de bloques de la codificación HEVC se muestra en la Figura 4.1.

HEVC también introduce importantes mejoras en los módulos de predicción intra, una nueva predicción adaptativa de los parámetros que indican el movimiento, un nuevo filtro de reducción de artefactos de bloque y una nueva versión optimizada de CABAC.

En los siguientes apartados se explicará brevemente la funcionalidad de los distintos módulos que componen el codificador y sus principales diferencias respecto a su predecesor H.264.

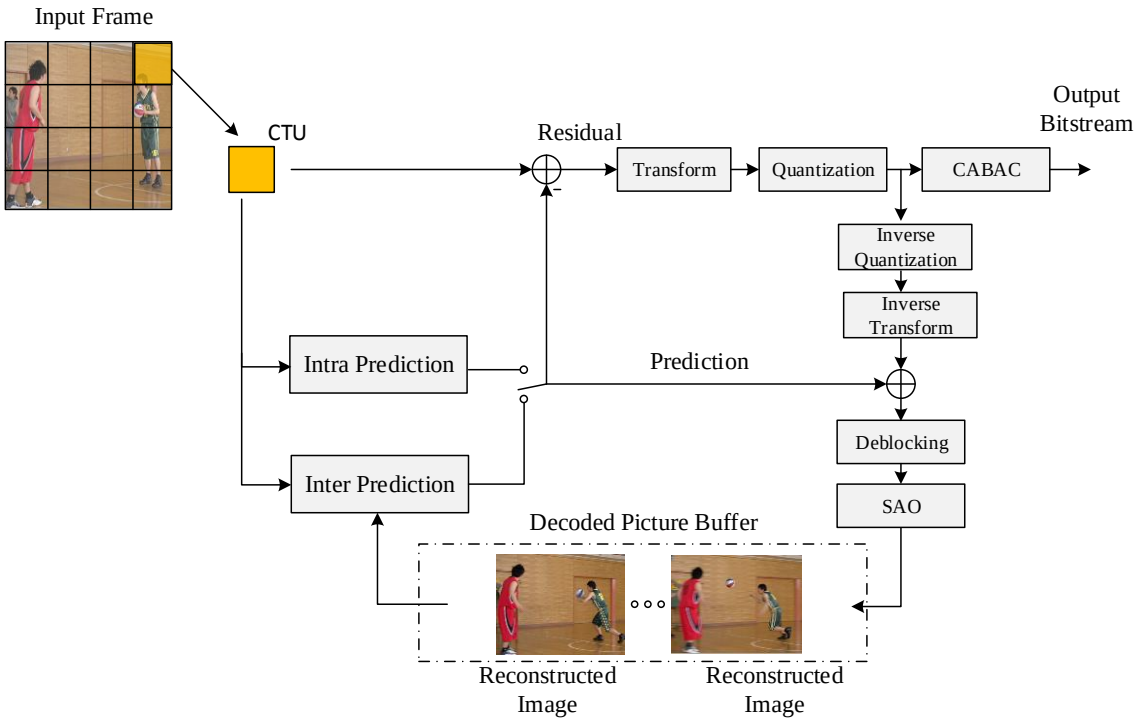


Figura 4.1. Diagrama de bloques de un codificador HEVC.

4.2 División de la imagen

En HEVC cada imagen es dividida en *tiles* y/o *slices*. Slices y/o tiles son divididos en las unidades básicas de codificación denominadas *Coding Tree Unit (CTU)*. Las slices son estructuras de datos que pueden ser descodificadas independientemente de otras slices de la misma imagen. Las slices no tienen por qué ser necesariamente rectangulares. Una slice está constituida a su vez por varios *slice segments*, comenzando con un segmento independiente y un conjunto posterior de segmentos dependientes. El segmento independiente está constituido por elementos de sintaxis que no son inferidos de los valores de elementos de sintaxis de otros segmentos. Tile es una región rectangular de la imagen constituida por varias CTU, que a su vez pueden estar contenidas en varias slices. De forma análoga, una slice está constituido por CTU que pueden estar dispersos por varios tiles. Por lo tanto, dentro de la misma imagen pueden coexistir slices que contengan varios tiles, y tiles que contengan varias slices. En la Figura 4.2, se muestra un ejemplo de división de la imagen en tiles y en slices. En el ejemplo indicado, una imagen constituida por 64 CTU se ha dividido en dos tiles de 4x8 CTU cada uno. El primer tile (Tile0) contiene a su vez una slice (Slice0), constituida por un segmento independiente de dos CTU y otro segmento dependiente de 30 CTU. El segundo tile (Tile1) contiene en su interior a dos slices, la primera slice (Slice 1) está formada por un segmento independiente de 3 CTU y otros dos segmentos dependientes. La segunda slice (Slice 2), contiene un segmento independiente de 2 CTU y otro dependiente de 8 CTU.

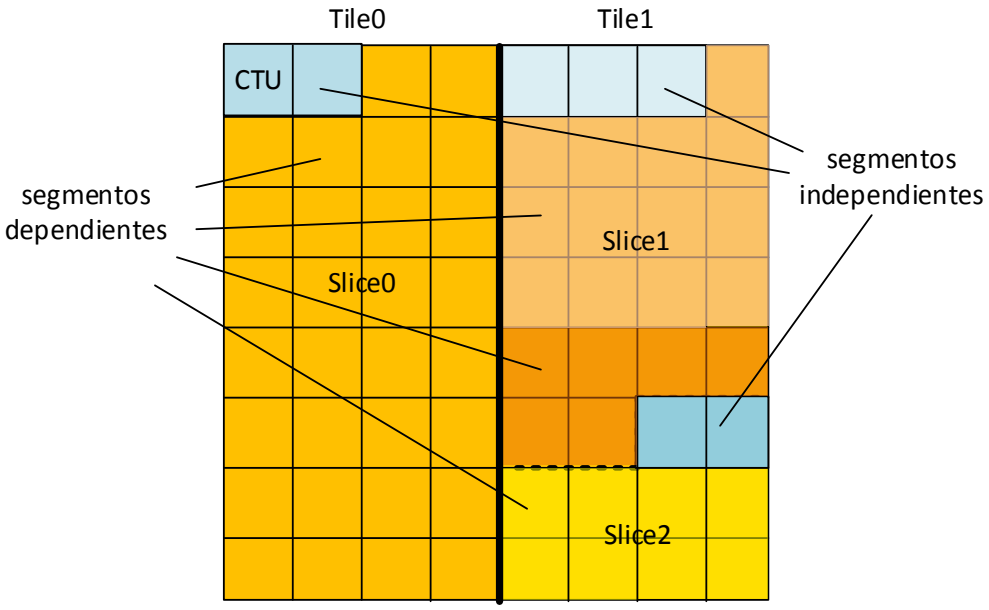


Figura 4.2. División de una imagen en tiles y slices.

El concepto de CTU es análogo al de macrobloque en estándares anteriores, pero a diferencia de H.264 donde sólo se permitía el uso de macrobloques de 16x16 píxeles, en HEVC se permite el uso de tamaños de CTU de 64x64, 32x32 y 16x16 píxeles. A su vez, una CTU es dividida en regiones cuadradas conocidas como *Coding Unit (CU)* de tamaños 64x64, 32x32, 16x16 y 8x8 píxeles. Cada CU es configurada para usar un tipo de predicción en particular, intra o inter, y es representada como un nodo dentro de la estructura de división quad-tree. Una CU puede ser dividida en una, dos o cuatro *Prediction Unit (PU)* dependiendo de su modo de partición. Los tamaños permitidos en una PU varían entre 64x64 píxeles y bloques de 4x4. Para una CU de tipo intra, sólo se permiten PU cuadradas mientras que una CU de tipo inter puede presentar los tipos de partición que se indican en la Figura 4.3. Los cuatro últimos modos de partición que aparecen en la Figura 4.3 se conocen como modos asimétricos (A*symmetric Motion Partitioning, AMP*). El tamaño 4x4 (PART_NxN) en PU tipo inter no es permitido para reducir los requerimientos de ancho de banda en el uso de la memoria en la compensación de movimiento. Al contrario que en el caso de intra, las PU inter pueden utilizar los tamaños 4x8 y 8x4.

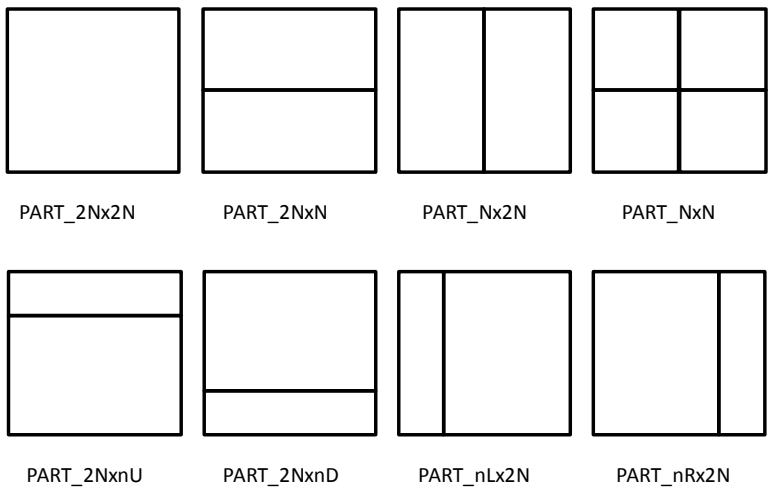


Figura 4.3. Modos de partición inter.

Por lo tanto, el proceso de particionado de la imagen comienza a nivel de CTU y termina en el mínimo tamaño de CU permitido o cuando no se requiera continuar debido a que el coste RDO del uso de unidades de codificación de tamaño inferior supere al coste asociado a los bloques de tamaño superior por las propias características del vídeo.

Por otro lado, existe el concepto de unidades básicas de transformación conocidas como *Transfom Unit (TU)* cuyos tamaños varían entre 32x32, 16x16, 8x8 y 4x4, siendo siempre bloques cuadrados al ser así requerido por las transformadas que se utilizan durante el proceso de codificación.

En la predicción inter, la predicción de la PU que se codifica se obtiene utilizando imágenes de referencia codificadas en instantes anteriores, explotando la redundancia temporal. Adicionalmente cada PU tiene asociado un conjunto de parámetros que describen su movimiento respecto a las referencias empleadas. En el caso de la predicción intra, la predicción de cada PU es obtenida utilizando píxeles reconstruidos de las PU vecinas.

Las componentes de color son procesadas por separado, dividiendo una CU en *Coding Blocks (CB)*, por lo que dentro de una CU habrá un CB para la luminancia y un CB para cada componente de color, existiendo diferente número de CB en función del tipo de muestreo de color utilizado. Análogamente, una PU es dividida en *Prediction Blocks (PB)* y las TU, en diferentes *Transform Blocks (TB)*.

Esta estructura de división de la imagen es muy versátil y permite adaptarse al contenido del vídeo a codificar. Los tamaños máximo y mínimo de la CTU pueden ser configurados como parámetro de entrada al codificador.

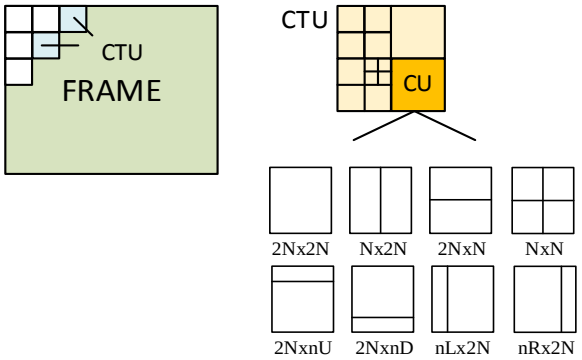


Figura 4.4. Quad-tree coding block partitioning.

Un concepto utilizado dentro del proceso de división recursivo es el de profundidad (*depth*), que indica cada uno de los diferentes estadios de división respecto al máximo tamaño permitido de CTU. Así pues, si el tamaño de CTU es 64x64, el tamaño máximo permitido de CU es 64x64, y cada nivel de profundidad indicará que se está evaluando un diferente tamaño de CU. En el ejemplo que se muestra en la Figura 4.5, depth=0 corresponde con una CU de tamaño 64x64, depth=1 a 32x32, depth=2 a 16x16 y depth=3 al tamaño 8x8. Si el tamaño de CTU fuese 32x32, entonces depth=0 corresponde a la CU de tamaño 32x32, depth=1 al estadio referente a las cuatro CU de tamaño 16x16, y depth=2 a las de tamaño 8x8; no existiendo la posibilidad de evaluar depth=3. En la Figura 4.6 se indica el orden en el que son procesadas las distintas CTU que componen la imagen (*raster scan*) y el orden en el que se procesan las distintas CU/PU/TU dentro de una misma CTU.

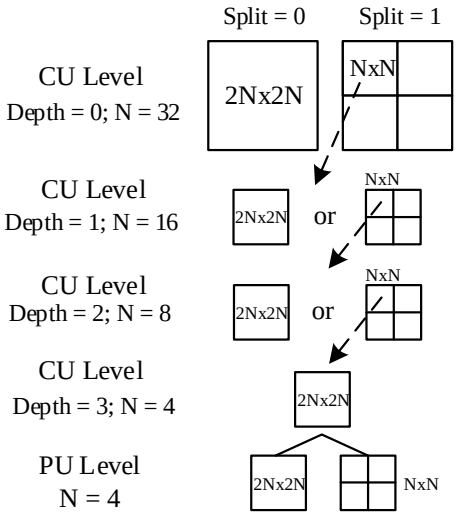


Figura 4.5. Proceso de división recursiva de una CU 64x64 tipo intra.

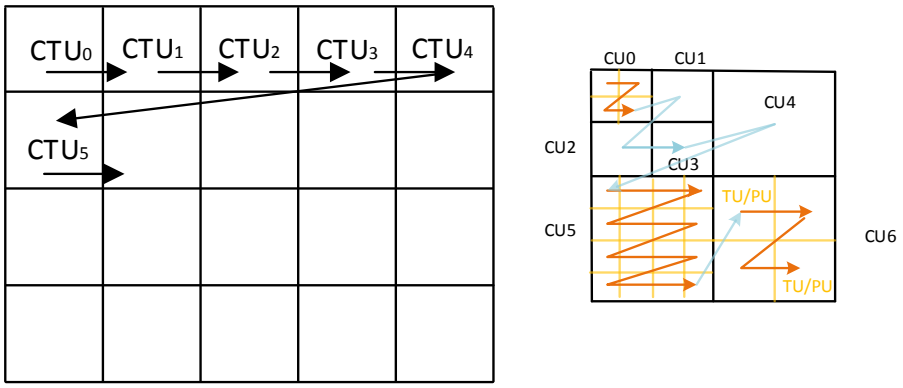


Figura 4.6. Orden de procesado en la división recursiva.
Orden de procesado de CTU (izquierda) y orden de procesado CU/PU/TU dentro de una CTU (derecha).

4.3 Predicción intra

En HEVC existen 35 modos de predicción intra: DC, plano y 33 modos direccionales (o angulares). En la Figura 4.7 se muestran los modos de predicción intra y la dirección asociada. La numeración mostrada para los modos de predicción es la indicada por el estándar.

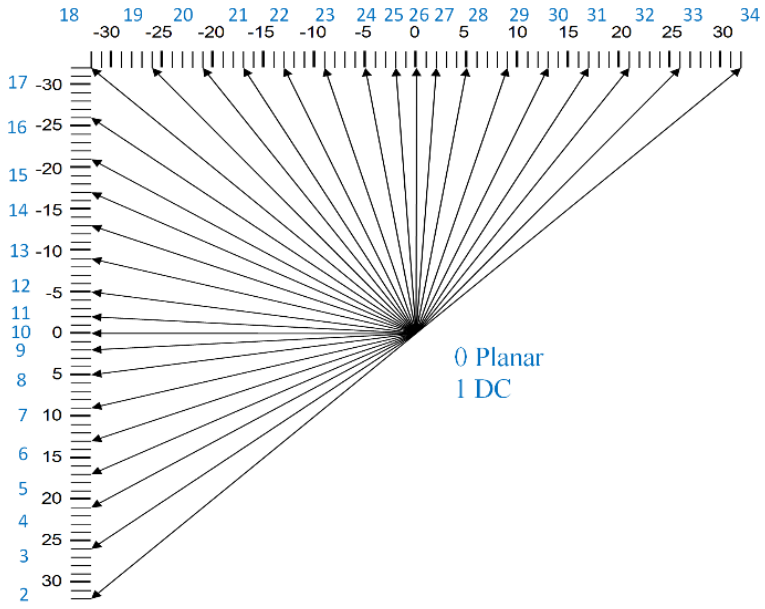


Figura 4.7. Relación entre las direcciones de predicción (en color negro) y los modos de predicción intra asociados (azul).

Los diferentes modos de predicción intra permiten obtener las muestras de predicción para un determinado PB mediante interpolación de muestras previamente reconstruidas en el canal de la luminancia, o del color que se esté considerando. En el caso de PB asociados a la crominancia, existe la posibilidad de aplicar el modo Plano, DC, horizontal, vertical o el mismo modo que se aplicó al PB correspondiente de la luminancia. Un plano en el que sólo se aplica predicción intra y que puede ser descodificado sin hacer uso de planos previamente codificados es conocido como I-frame, o en el caso de un slice, como *I-slice*.

Respecto a H.264, el estándar HEVC incrementa considerablemente el número de modos de predicción intra que se pueden utilizar. Los modos de predicción intra disponibles en H.264 son los siguientes:

1. En bloques 4x4: DC + 8 modos direccionales,
2. en bloques 16x16: DC + Plano + Vertical + Horizontal,
3. para la crominancia: DC + Plano + Vertical + Horizontal.

Como se comentó con anterioridad, el conjunto de modos de predicción intra utilizados por el estándar HEVC amplía a 33 los modos direccionales o angulares. El conjunto de modos de predicción intra en HEVC abarca todos los disponibles en H.264 y añade la posibilidad de usar otras direcciones adicionales, tal y como se indica en la Figura 4.8.

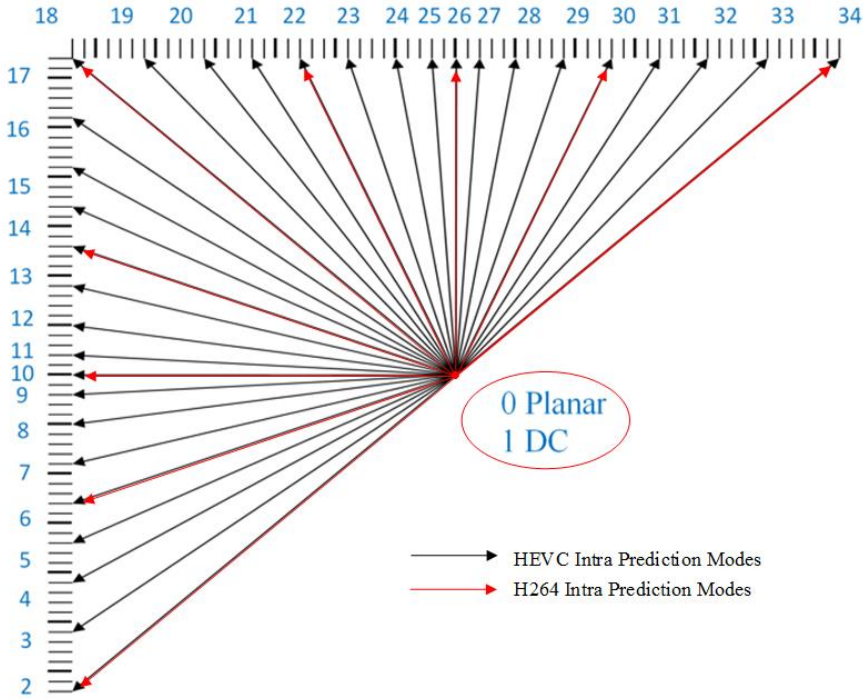


Figura 4.8. Correlación entre modos de predicción intra en H.264 y HEVC.

Como se puede observar en la Figura 4.8, se han añadido modos de predicción adicionales en las inmediaciones de los modos 10 (horizontal) y 26 (vertical) al ser éstos los modos más probables de ser utilizados, estadísticas obtenidas a partir de la utilización de H.264. Respecto a H.264, en HEVC se introduce la mejora de poder indicar como modo de predicción intra aplicado a la crominancia el mismo que se utilizó en el bloque equivalente de luminancia.

HEVC añade en la luminancia, el filtrado del conjunto de píxeles vecinos que se utilizarán como referencia para obtener la predicción. La fortaleza del filtro es controlada dependiendo del modo de predicción intra a utilizar y del tamaño del TB. El objetivo del filtrado es eliminar artefactos en los contornos de los objetos, provocados por la existencia de bordes en el array de píxeles de referencia. Dicho filtrado no es aplicado en el caso del modo DC o si el TB es de tamaño 4x4. En H.264 se aplicaba un filtrado similar pero sólo para macrobloques de tipo intra 8x8.

HEVC también añade la posibilidad de realizar un filtrado durante el proceso de reconstrucción de TB tipo intra para reducir las discontinuidades en las fronteras entre distintos bloques. Este filtro conocido como *Intra-Boundary Filter (IBF)*, puede ser utilizado cuando se predicen muestras a lo largo de los bordes izquierdo y/o superior para los modos de predicción DC, horizontal y vertical.

En la Figura 4.9, se indican las muestras vecinas utilizadas para obtener la predicción intra de un bloque de tamaño NxN píxeles.

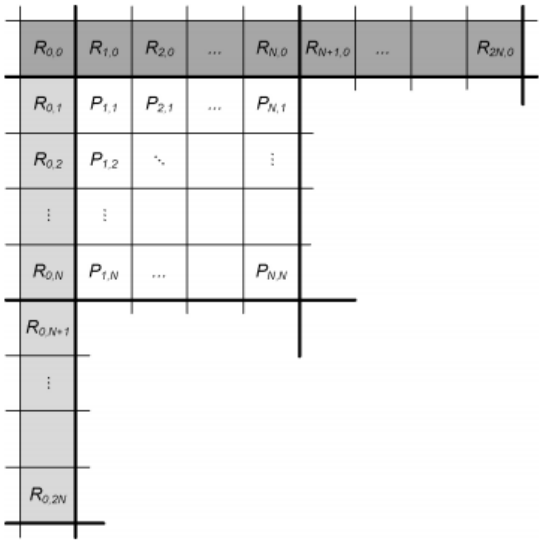


Figura 4.9. Nomenclatura de las muestras vecinas (Rx, y) requeridas para la predicción intra (Px, y) de un bloque NxN.
Imagen extraída de [1].

En determinadas posiciones de la imagen, las muestras vecinas pueden no estar disponibles debido a que se esté trabajando en los bordes de la imagen o en las fronteras entre distintos slices. En estos casos, las muestras son derivadas siguiendo un proceso definido en el estándar. Por ejemplo, en el caso de no estar disponibles los vecinos de la izquierda, se generarán por repetición de las muestras de referencia más cercanas del PB situado por debajo, o si éste no estuviese tampoco disponible, empleando las muestras del PB superior. De forma similar, si las muestras superiores no están disponibles, se sustituirán por una copia de las muestras más cercanas del PB de la izquierda. Si no estuviese ningún vecino disponible se asignará a las muestras un valor prefijado que depende de la precisión utilizada para definir un píxel (128, si se usan 8 bits). A diferencia de H.264, HEVC incorpora como muestras vecinas las pertenecientes al PB del vecino situado en la parte inferior izquierda (en la Figura 4.9 se indican como $R_{0,N+1}…R_{0,2N}$), debido a que la estructura de división de la imagen utilizada en HEVC permite que esté disponible más frecuentemente que en H.264.

La Ecuación 4.1 describe la fórmula genérica aplicada para obtener la predicción en los modos angulares. En la Ecuación 4.2 se muestra la correspondiente al modo Plano. Como se puede observar el modo Plano es una media de dos predicciones lineales.

$$P_{x,y} = \left((32 - w_y) * R_{i,0} + w_y * R_{i+1,0} + 16\right) \gg 5\ , \tag{4.1}$$
$$c_y = (y * d) >> 5\ ,$$
$$w_y = (y * d)\ \&31\ ,$$
$$i = x + c_y.$$

$$P^V_{x,y} = (N - y) * R_{x,0} + y * R_{0,N+1}\ , \tag{4.2}$$
$$P^H_{x,y} = (N - x) * R_{0,y} + x * R_{N+1,0}\ ,$$
$$P_{x,y} = \left(P^V_{x,y} + P^H_{x,y} + N\right) \gg (log_2(N) + 1\ ).$$

En HEVC para reducir la complejidad del proceso de predicción intra, las muestras de referencia de un PB son proyectadas a una única fila o columna dependiendo de la

direccionalidad del modo de predicción utilizado. Se utiliza la columna izquierda para los modos 2 a 17 y la fila de arriba, para los modos 18 a 34. En algunos casos las posiciones proyectadas podrían tener valores negativos, en tales casos se extiende la fila por proyección de la columna de la izquierda (para la referencia de la fila de arriba) o se extiende la columna por proyección de la fila de arriba (para la referencia de la columna de la izquierda). En la Figura 4.10 se muestra un ejemplo de la proyección de la columna vecina izquierda para extender la fila vecina superior.

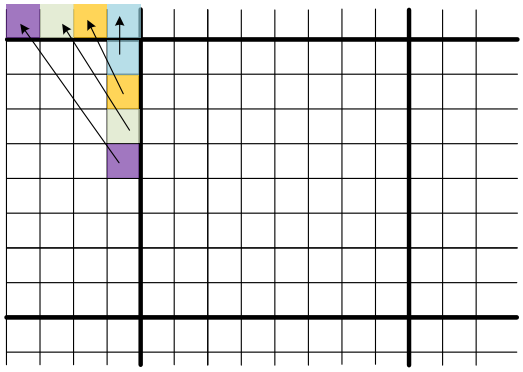


Figura 4.10. Ejemplo de proyección de la referencia izquierda sobre la superior.

Según la Ecuación 4.1, en los modos angulares una muestra $P_{x,y}$ es obtenida como proyección de su posición en una fila de píxeles aplicando la dirección de predicción asociada al modo correspondiente e interpolando utilizando una precisión de píxel de 1/32 (operación desplazamiento >>5). w_y es una ponderación de dos píxeles de referencia correspondiente a la posición en la proyección entre $R_{i,0}$ y $R_{i+1,0}$. El índice para las muestras de referencia i y w_y, son calculados a partir del desplazamiento de la proyección d asociada al modo de predicción utilizado [1]. La Tabla 4.1 indica los valores de d para la direccionalidad vertical (V) y la horizontal (H).

Tabla 4.1. Valores del parámetro de desplazamiento 'd' en función del modo de predicción intra utilizado.

Modo de predicción intra (horizontales)	Valor 'd'	Modo de predicción intra (verticales)	Valor 'd'
2	32	18	-32
3	26	19	-26
4	21	20	-21
5	17	21	-17
6	13	22	-13
7	9	23	-9
8	5	24	-5
9	2	25	-2
10	0	26	0
11	-2	27	2
12	-5	28	5
13	-9	29	9
14	-13	30	13
15	-17	31	17
16	-21	32	21
17	-26	33	26
		34	32

La Ecuación 4.1 muestra la obtención de la predicción en los modos angulares verticales (18 a 34) utilizando como referencia la fila vecina del bloque superior. Sustituyendo en dicha ecuación la coordenada *y* por *x*, se obtiene la fórmula para los modos horizontales (2 a 17) a partir de las muestras de referencia de la columna de píxeles de referencia de la izquierda. Cuando w_y o w_x toma el valor 0 (modo vertical 26 u horizontal 10) las muestras de la predicción se obtienen como copia directa de las muestras de referencia. En el modo DC, la predicción se obtiene por medio de una media de las muestras vecinas.

4.4 Predicción inter

Cada PU tipo inter tiene asociado un conjunto de parámetros que consisten en uno o dos vectores de movimiento, índices a la imagen de referencia utilizada y un indicador de la lista de imágenes de referencia utilizada. Estos parámetros posibilitan la descodificación de la PU mediante la generación de las muestras correspondientes a la predicción. Dichos parámetros pueden señalizarse de una forma implícita o explícita.

El modo *skip* es utilizado en una CU que posee únicamente una PU que no contiene coeficientes significativos y además, ni se codifican los vectores de movimiento ni el índice de referencia asociado. El modo *merge* es utilizado cuando la información del movimiento asociado a la PU puede ser derivado completamente de vecinos espaciales y temporales, que son obtenidos mediante un proceso indicado en el estándar. Por lo tanto, cuando se utiliza el modo merge o el modo skip no se envía información relativa al movimiento de la PU suponiendo su uso, un importante ahorro de bits. En el resto de modos de predicción inter, se envía al descodificador la información relativa al movimiento de la correspondiente PU de una forma explícita.

Si únicamente se utiliza una lista de referencias para obtener la predicción, se conoce como predicción unidireccional (*uni-prediction)*. Este tipo de predicción es la única que se permite en slices tipo P o *P-slices*. Cuando para obtener la predicción se han utilizado las dos listas disponibles de referencias, la predicción de la PU es producida a partir de dos bloques de muestras de referencias. Este tipo de predicción se conoce como predicción bidireccional (*bi-prediction*).

4.4.1 Derivación de los vecinos en el modo merge

En el modo merge la información de movimiento se deriva exclusivamente a partir de los vecinos espaciales y/o temporales. Los diferentes vecinos disponibles conforman una lista de candidatos que son evaluados. El proceso de elaboración de la lista de candidatos está establecido dentro del estándar de forma que finalmente, lo que se codifica y envía al descodificador es el índice que indica la posición del vecino finalmente escogido dentro de la lista de posibles candidatos. La Figura 4.11 resume el proceso de obtención de la lista de candidatos.

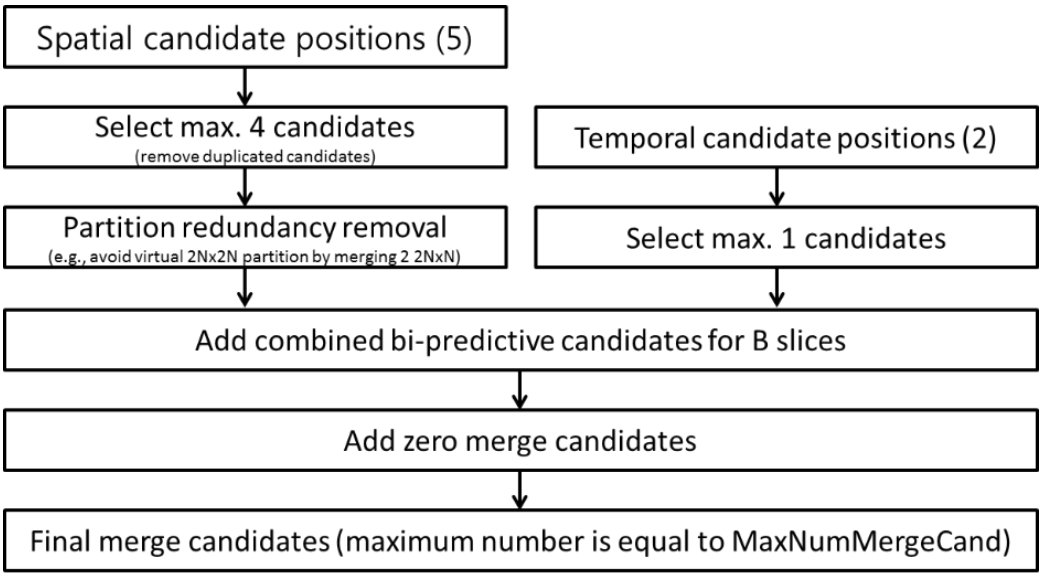


Figura 4.11. Proceso de derivación de los candidatos en el modo merge. Imagen extraída de [2].

Hasta un máximo de 4 candidatos espaciales pueden ser utilizados. Los candidatos espaciales para una determinada PU son derivados según su disponibilidad siguiendo el orden $A_1 \rightarrow B_1 \rightarrow B_0 \rightarrow A_0 \rightarrow B_2$. La posición respecto a la PU actual que corresponde a la anterior nomenclatura se indica en la Figura 4.12:

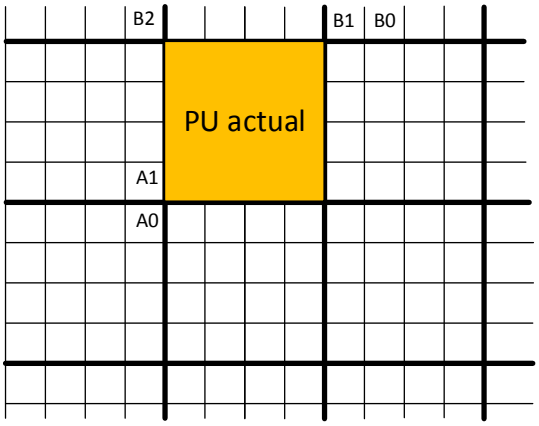


Figura 4.12. Candidatos derivados de los vecinos espaciales de la PU actual en el modo merge.

En la Figura 4.13 se representan los vecinos disponibles en el caso de la segunda PU de particiones tipo Nx2N, pero también es aplicable en los casos nLx2N y nRx2N. En esos casos, el orden de derivación es el siguiente, $A_1 \rightarrow B_0 \rightarrow A_0 \rightarrow B_2$. En la Figura 4.14 se representan los posibles vecinos para la segunda partición en los casos 2NxN, 2NxnU y 2NxnD; siendo el orden de derivación $A_1 \rightarrow B_0 \rightarrow A_0 \rightarrow B_2$.

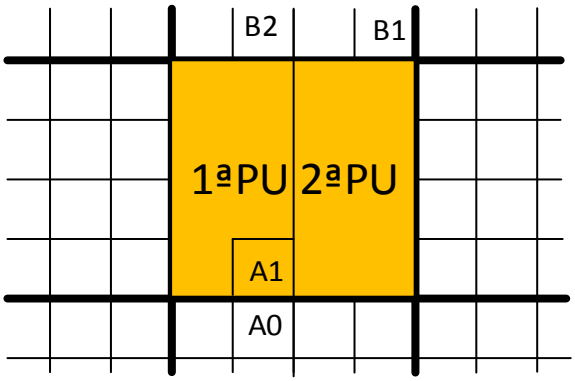


Figura 4.13. Candidatos derivados de los vecinos espaciales en el modo merge para la segunda PU empleando el tipo de partición Nx2N.

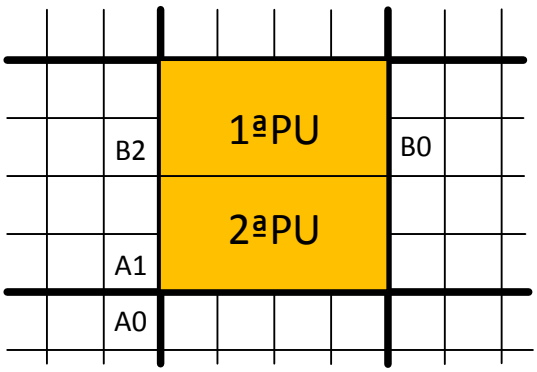


Figura 4.14. Candidatos derivados de los vecinos espaciales en el modo merge para la segunda PU empleando el tipo de partición 2NxN.

Como se puede observar, existe una dependencia entre los bloques espaciales vecinos a la hora de obtener la lista de candidatos. Para eliminar esta dependencia y posibilitar el cómputo de la estimación de movimiento en paralelo, los candidatos situados dentro de la misma *Merge Estimation Region (MER)* no son incluidos en la lista. Por lo tanto, MER es una región en la cual los candidatos en el modo merge pueden ser derivados de forma totalmente independiente. El tamaño de la MER es indicado mediante el correspondiente campo del bitstream para que el equipo descodificador disponga de tal información. Los tamaños posibles son 4x4 (no hay posibilidad de aplicar paralelismo), 8x8, 16x16, 32x32 y 64x64.

El vector de movimiento del candidato temporal es obtenido accediendo a la PU co-situada (*co-located PU*). Siendo esta PU la situada en las posiciones C3 y H que se indican en la Figura 4.15, en la imagen de referencia más cercana siguiendo el orden de presentación. C0 indica las coordenadas equivalentes a la PU actual en la imagen de referencia utilizada. Si la PU vecina situada en la posición H no está disponible, es tipo intra o está fuera de la CTU actual, entonces se utiliza la posición C3.

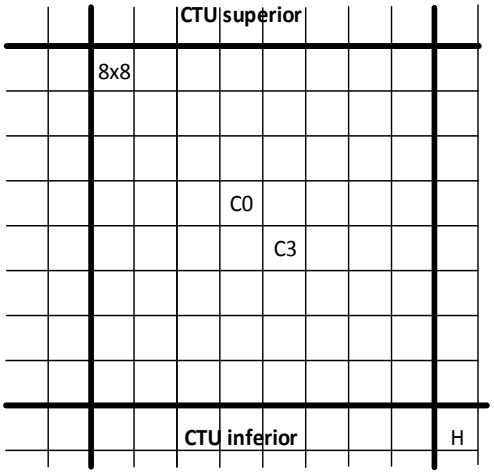


Figura 4.15. Ejemplo de PU co-situadas empleando CTU de tamaño 64x64.

Sobre el vector de movimiento del candidato temporal se realiza un escalado cuyo proceso de derivación es ilustrado en la Figura 4.16. Para obtener el candidato temporal es necesario el uso de un contador que indica el orden de presentación asociado a cada imagen, conocido como *Picture Order Count (POC)*. En la Figura 4.16, el parámetro *tb* define la diferencia entre el POC de la imagen de referencia que utiliza la imagen actual y el POC de la imagen actual. Por otro lado, *td* define la diferencia entre el POC de la referencia de la imagen donde se sitúa la PU co-situada y su propio POC. De este proceso, se derivan únicamente los vectores de movimiento ya que el índice de la imagen de referencia del candidato temporal siempre es cero.

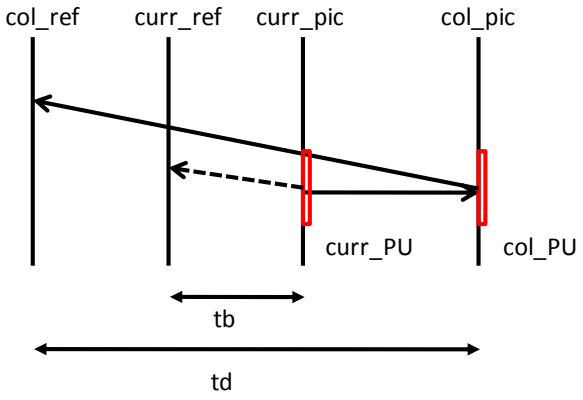


Figura 4.16. Proceso de derivación del candidato temporal en el modo merge. Imagen extraída de [2].

Cuando se está procesando un slice tipo B, los candidatos espaciales y temporales derivados como se ha indicado previamente, son combinados para realizar predicciones bi-direccionales utilizando ambas listas de referencia. Tanto para slices tipo P como B, el vector de movimiento de coordenadas (0, 0) siempre es añadido a la lista de candidatos.

4.4.2 Derivación de los vecinos en el resto de modos

De una forma similar a la indicada para el modo merge, para el resto de modos de predicción se crea una lista de candidatos que permitan obtener una predicción del vector del movimiento explotando la correlación de la PU actual con sus vecinos espaciales y temporales. Se debe recordar que los vectores de movimiento obtenidos por la estimación de movimiento no se codifican para su posterior envío directamente, si no que se codificará la diferencia del vector obtenido respecto al vector de movimiento predicho cuyo proceso de derivación es el que se está describiendo. En la Figura 4.17, se indica el proceso a realizar para derivar la predicción del vector de movimiento según el estándar.

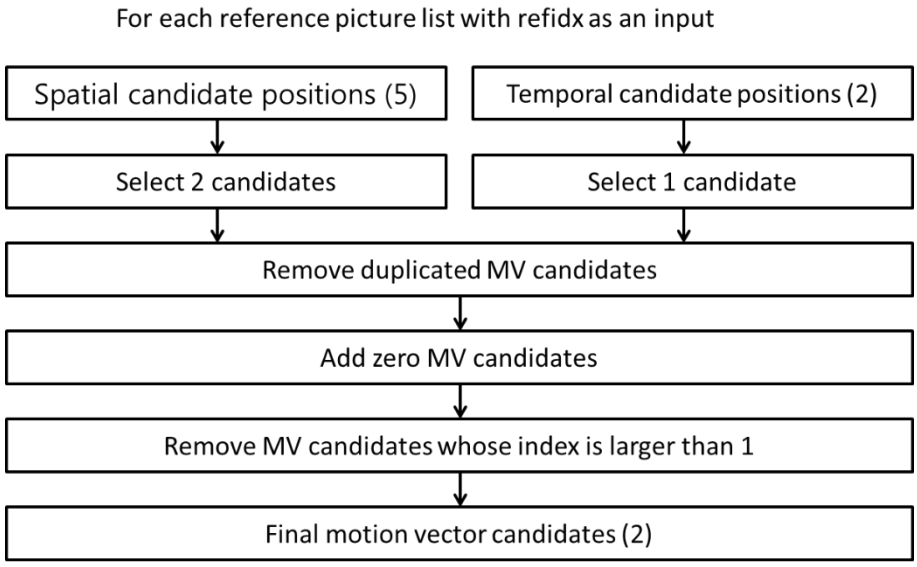


Figura 4.17. Proceso de derivación de los candidatos en modos inter (a excepción del modo merge).
Imagen extraída de [2].

Los cinco candidatos espaciales son derivados utilizando las posiciones indicadas en las Figuras 4.12, 4.13 y 4.14 para el modo merge. A los vectores de movimiento de los vecinos candidatos se les puede aplicar un escalado en función de que usen o no la misma lista de referencias que la PU a codificar y de que tengan o no el mismo POC. El orden de derivación a aplicar para el vecino de la izquierda es $A_0 \rightarrow A_1 \rightarrow A_{0\ escalado} \rightarrow A_{1escalado}$ y $B_0 \rightarrow B_1 \rightarrow B_2 \rightarrow B_{0\ escalado} \rightarrow B_{1\ escalado} \rightarrow B_{2\ escalado}$ para el caso del vecino superior. El proceso de derivación de los candidatos temporales es el mismo que en el caso del modo merge, a excepción de que el índice de la imagen de referencia del candidato temporal puede ser distinto de 0 y de que es enviado al descodificador.

4.4.3 Interpolación

HEVC especifica en la luminancia vectores de movimiento de precisión máxima de un cuarto de píxel, pero utiliza para realizar la interpolación un filtro de 8 etapas frente a las 6 etapas utilizadas por H.264. Para la crominancia utiliza vectores de 1/8 de píxel empleando un filtro de interpolación de 4 etapas, lo cual mejora considerablemente el filtro bilineal utilizado en H.264. Esta mejora a nivel de sub-píxel en la precisión del filtrado

incrementa la eficiencia en la estimación/compensación de movimiento pero también es una estructura más exigente en términos de accesos a memoria y potencia computacional. Por ello, en HEVC se decidió no permitir particiones inter 4x4, limitando el mínimo tamaño de PU a 4x8 y 8x4 en el caso de predicción unidireccional, y a 8x8 en el caso de la predicción bidireccional.

4.5 Transformación y cuantificación

Como se indicó previamente, en HEVC el tamaño de TB de luminancia ha de ser siempre cuadrado y puede tomar los valores 64x64, 32x32, 16x16, 8x8 o 4x4; mientras que en H.264, el tamaño de TB de luminancia utilizado era 4x4 u 8x8. Otra diferencia respecto a H264, es que en HEVC no se permite la transformada de tamaño 2x2 para los TB de crominancia, siendo el tamaño de TB mínimo 4x4.

En HEVC, también se proporciona la posibilidad de que la transformación no sea realizada (*Transform Skip, TS*) o la posibilidad de que un TB sea codificado sin pérdidas (*lossless mode*). Si no se utiliza ninguno de los anteriores modos, la transformación es aplicada tal y como se indica en la Figura 4.18.

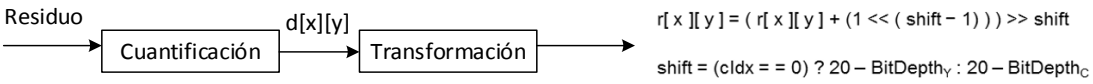


Figura 4.18. Proceso de cuantificación (scaling) y transformación en HEVC.

Si TS es utilizado, entonces la operación de transformación es sustituida por un desplazamiento de bits tal y como se muestra en la Figura 4.19.

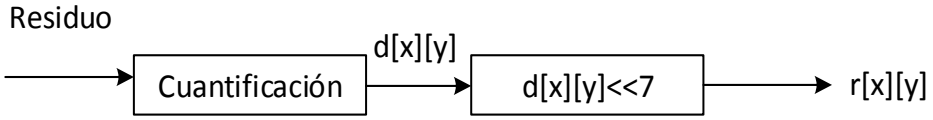


Figura 4.19. Proceso de cuantificación y transformación en HEVC aplicando TS.

Si el modo de codificación sin pérdidas es utilizado, entonces el proceso de cuantificación y transformación queda definido por la Ecuación 4.4.

$$r[x][y] = Residuo[x][y]. \quad (4.4)$$

HEVC utiliza las transformadas DCT y DST. DST es utiliza únicamente a nivel de TB tipo intra 4x4 debido a que se observaron en una serie de experimentos una leve mejora en la tasa de compresión respecto al uso de DCT. DCT es utilizada en el resto de tamaños de TB.

4.6 Codificación entrópica

HEVC utiliza CABAC y VLC para convertir los elementos de sintaxis en bits para su posterior transmisión o almacenamiento. VLC es utilizado para las cabeceras a nivel de slice y otros datos que no requieran de una tasa de compresión muy elevada, debido a que el número de bits que generan es muy pequeño en comparación al total de bits requeridos en la compresión de la secuencia de vídeo.

CABAC alcanza altas tasas de compresión por medio de la selección de modelos de probabilidad para cada elemento de sintaxis conforme a un contexto asociado. La estimación de la probabilidad de ocurrencia de un símbolo es continuamente adaptada basándose en estadísticas locales y utilizando codificación aritmética.

Cada elemento de sintaxis es convertido en un conjunto de *bins* mediante un proceso conocido como *binarization*. Mediante este proceso, valores de símbolo distintos a 0 o 1, como puede ocurrir en un coeficiente o vector de movimiento, son convertidos a un código binario. Los bins pueden tener asociados un modelo de contexto (*context bins*) o ser de tipo *bypass*. El modelo de contexto indica el valor de la probabilidad de uno o varios bins dentro de un símbolo binario. El coste de codificar un bin con contexto asociado depende directamente de la probabilidad de que el bin que se debe codificar coincida con el bin considerado como más probable.

La codificación aritmética es la encargada de codificar cada bin conforme el modelo de probabilidad seleccionado. En el caso del CABAC, la codificación aritmética sólo utiliza dos sub-rangos para cada bin, correspondientes a los valores 0 y 1. La codificación de estos bins provoca una actualización del contexto de probabilidad correspondiente al valor finalmente codificado, lo que provoca que los bins que utilicen el mismo contexto deban ser ejecutados en un proceso en serie. Siendo un proceso que se ejecuta a escala binaria, se trata de un proceso computacionalmente muy costoso. Por otro lado, los bins tipo bypass no requieren de actualización de contexto de probabilidad y tienen asociado un proceso de codificación simplificado. Estas condiciones posibilitan que los bins tipo bypass puedan ser rápidamente codificados en paralelo, lo que incrementa considerablemente el número de bins que pueden ser procesados en un segundo en el CABAC. Teniendo en cuenta la ventaja computacional que aporta este tipo de bins, en HEVC el codificador CABAC fue diseñado para posibilitar la generación de mayor número de bins tipo bypass que su predecesor. Este diseño fue en parte influido por los problemas de implementación que se sufrieron en el CABAC de H.264 en el ámbito de desarrollo de codificadores comerciales.

4.7 Deblocking

Al igual que sucede en estándares de compresión de vídeo anteriores, HEVC realiza la codificación dividiendo la imagen en bloques y aplicando sobre cada bloque una transformada y cuantificación de forma independiente. Eso provoca una pérdida de correlación entre los distintos bloques y la aparición de discontinuidades en los bordes de los bloques, por lo que las imágenes reconstruidas sufren lo que se denomina efecto de bloque. El filtro de deblocking mejora la calidad visual de las imágenes reconstruidas reduciendo tal efecto. Al estar incluido este filtro en el bucle de reconstrucción, las imágenes filtradas son utilizadas como referencias, y al presentar las referencias una mayor

calidad, la estimación de movimiento se realiza con mayor eficacia, provocando un ahorro en la tasa de datos generada. Se trata de un proceso establecido en el estándar y que debe seguirse minuciosamente para no provocar desajustes entre la imagen reconstruida producida por el codificador y la equivalente en el equipo descodificador. Son filtradas tanto la luminancia como las componentes de la crominancia.

El filtrado se realiza a nivel de bloques 8x8, a diferencia de H.264 donde se trabaja a nivel 4x4, lo que reduce considerablemente la carga computacional sin una pérdida apreciable de calidad. Primero se aplica un filtrado horizontal a toda la imagen para eliminar los bordes verticales, trabajando con líneas de 8 píxeles. Posteriormente se realiza un filtrado vertical para eliminar los bordes horizontales utilizando columnas de 8 píxeles, tal y como se indica en la Figura 4.20. El hecho de aplicar cada filtrado a toda la imagen permite el cómputo en paralelo.

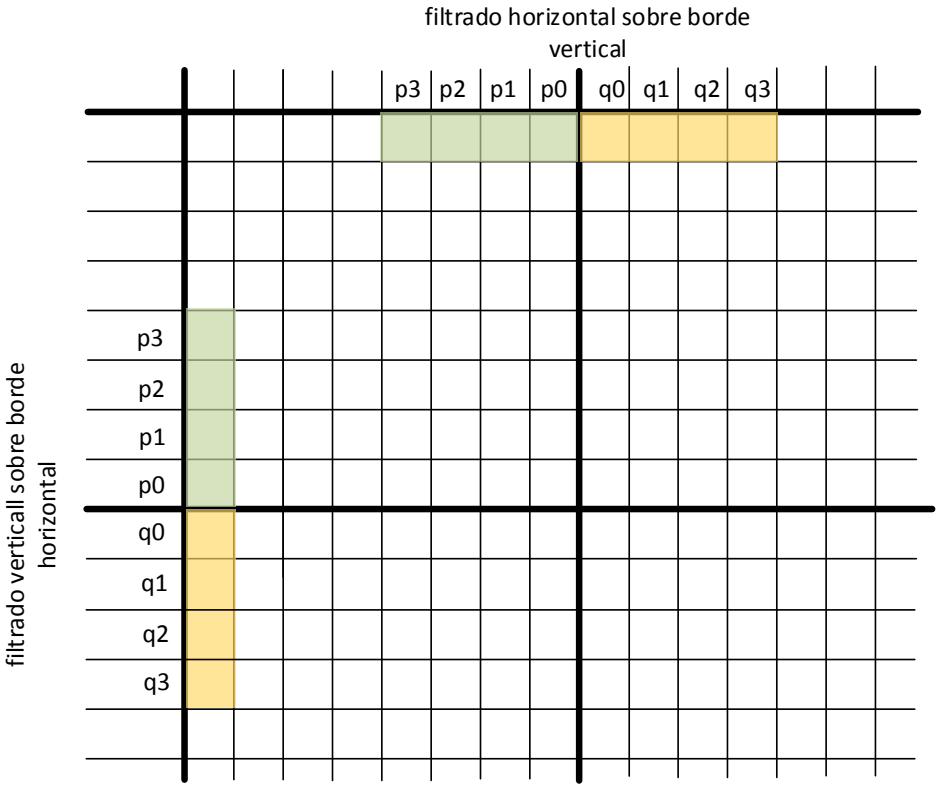


Figura 4.20. Filtro de deblocking.

El conjunto de condiciones a evaluar que determinan la fortaleza del filtrado (BS) se indican a continuación:

- BS=2; si uno de los dos bloques utiliza predicción intra.
- BS=1; si se cumple alguna de las siguientes condiciones:
  - Si p o q tiene al menos un coeficiente distinto de cero y se está filtrando una frontera entre diferentes TU.
  - La absoluta diferencia entre los vectores de movimiento de los bloques es >= 1 en unidades de píxel entero.
  - Los bloques tienen distinto índice de referencia.
  - El número de vectores de movimiento es distinto entre los dos bloques.
- BS=0 (no se realiza filtrado); si no se cumplen las condiciones anteriores.

Conforme el valor de BS y de QP, se obtienen dos umbrales: t_c y β que son obtenidos por medio de unas tablas predefinidas. Estos umbrales determinan si se debe aplicar un filtrado agresivo, uno débil o no filtrar. Los umbrales son compartidos por cada bloque 4x4 para reducir gasto computacional. Si el filtrado es muy agresivo se puede provocar un efecto de suavizado demasiado evidente que lleve a que la imagen sea borrosa. Por otro lado, un filtrado muy liviano provocará que se aprecie en demasía el efecto de bloque.

4.8 Sample Adaptive Offset (SAO) filter

Una de las novedades que incluye HEVC respecto a sus predecesores es la inclusión dentro del bucle de reconstrucción de un filtro conocido como *Sample Adaptive Offset (SAO)*. El principal objetivo que persigue SAO es la reducción de la distorsión introducida en los píxeles por la codificación. Esto se consigue por medio de una clasificación inicial de las muestras (*samples*) reconstruidas en diferentes categorías, obtener un correspondiente desplazamiento (*offset*) para cada categoría y posteriormente, añadir dicho desplazamiento a las distintas muestras de cada categoría. El desplazamiento para cada categoría es calculado en el codificador y correspondientemente señalado en el bitstream de una manera explícita para que el descodificador lo utilice.

La principal motivación de la inclusión de SAO se debe a que el uso de TU de tamaño grande puede llegar a introducir artefactos en los bordes de los objetos contenidos dentro de una TU (*ringing artifacts*). Estos artefactos son más evidentes a medida que se incrementa el valor de QP ya que provienen del error introducido en la cuantificación en las componentes de alta frecuencia en el dominio transformado [2]. SAO también mejora la agudeza visual percibida en los bordes de la imagen y elimina la aparición de pseudo-bordes (*banding artifacts*). En la Figura 4.21, se muestra una imagen en la que aparecen los artefactos comentados.

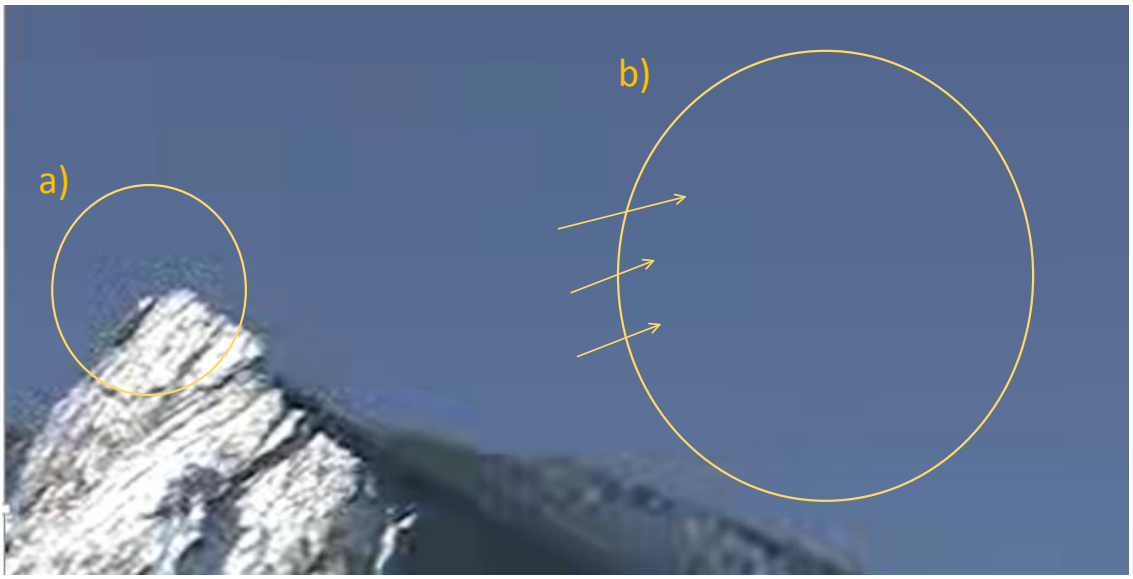


Figura 4.21. Artefactos debidos a la codificación
a) Ringing artifacts b) Banding artifacts.

SAO debe de ser aplicado tras el filtro de deblocking. La acción conjunta de ambos conlleva una mejora visual considerable, así como una reducción de tasa debido a la utilización de referencias para la predicción libres de artefactos. El desplazamiento se calcula consultando una tabla predefinida en función del gradiente local en la posición de la muestra a filtrar. Existen dos modos de SAO, *band offset* (*BO*) y *edge offset* (*EO*). En el modo EO, una vez que el CTB es clasificado en una determinada clase, las diferentes muestras son categorizadas en 5 distintas categorías. Las clases indican la orientación del gradiente utilizado, tal y como se indica en la Figura 4.22. En la Tabla 4.2 se indica la clasificación de las 5 posibles categorías.

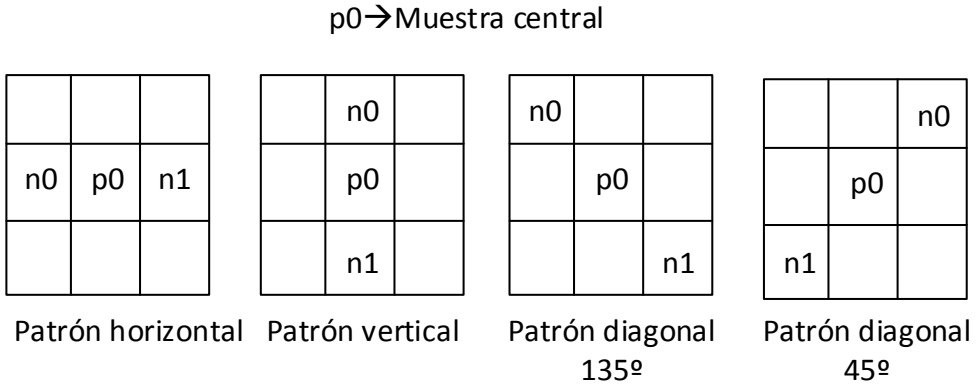


Figura 4.22. Patrón de gradiente utilizado para cada clase en SAO.

Tabla 4.2. Categorías modo EO.

EdgeIdx	Condición	Tipo de área
0	p=n0 and p=n1	Plana
1	p<n0 and p<n1	Mínimo local
2	(p<n0 and p=n1) or (p<n1 and p=n0)	Borde
3	(p>n0 and p=n1) or (p>n1 and p=n0)	Borde
4	p>n0 and p>n1	Máximo local

BO implica que el mismo desplazamiento sea sumado a todas las muestras de una misma banda. Cada banda contempla un conjunto de valores de amplitud. El rango posible de valores de una muestra es dividido en 32 bandas. Realmente, el mismo desplazamiento se aplica a cuatro bandas adyacentes debido a que está demostrado que en las zonas con texturas planas en la que los artefactos suelen ocurrir, la mayoría de amplitudes en un CTB tienden a estar concentradas en unas pocas bandas [2]. La tabla a consultar para el mejor patrón de gradiente y los cuatro desplazamientos son determinados por el codificador, y son señalados explícitamente o derivados respecto al CTB de la izquierda (indicándolo en el bitstream con el correspondiente flag).

4.9 Herramientas para aplicar paralelismo

HEVC ha sido diseñado para permitir la explotación del procesamiento de la imagen en paralelo, asignando la correspondiente tarea de codificación de una región de la imagen a un dispositivo de procesado. El uso de la división de la imagen en diferentes slices es una de las técnicas que permiten codificar varias áreas de la imagen al mismo tiempo, siempre y cuando cada slice sea independiente de las otras slices que se codifiquen en paralelo. El problema asociado al uso de varias slices independientes reside en que se reduce la explotación de las redundancias existentes, lo que afecta directamente a la tasa de compresión asociada. Por otro lado, a bajos bitrates se pueden apreciar artefactos visuales en las fronteras entre las diversas slices, ya sea por aplicar diferentes valores de QP porque el algoritmo de control de tasa no puede acceder a la información referente a las slices adyacentes o por una mala predicción debida a la no disponibilidad de vecinos espaciales. La utilización de slices ya estaba contemplada en el estándar H.264, por lo que la problemática asociada era ampliamente conocida cuando se definió HEVC. Por ello, se posibilitó el uso de una herramienta que permitiese la codificación en paralelo provocando una pérdida de calidad visual muy inferior. Esta nueva herramienta es conocida como *Wavefront Parallel Processing (WPP),* y consiste en la división de cada slice en filas de CTU. La primera fila es procesada de la manera tradicional, mientras que la segunda fila puede comenzar cuando se han procesado las primeras dos CTU de la primera fila. De forma similar, el resto de filas comienzan a procesarse cuando se han finalizado las dos primeras CTU de la fila anterior. La Figura 4.23 muestra el proceso seguido en la codificación WPP.

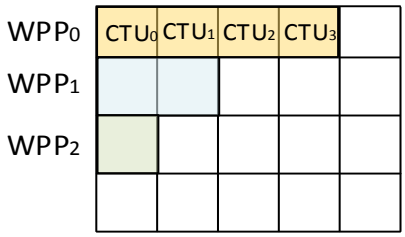


Figura 4.23. Wavefront Parallel Processing (WPP).

En estándares previos la predicción podía realizarse de una forma análoga a las WPP, pero no era posible trabajar siguiendo este orden en el codificador entrópico, por lo que HEVC ha posibilitado de una forma muy eficiente la aplicación del paralelismo en el proceso de codificación. Los modelos de contexto de cada fila son inicializados utilizando únicamente dos CTU de la fila anterior, lo que afecta mínimamente a la tasa de compresión, a diferencia de lo que ocurre utilizando slices y/o tiles. De forma equivalente, la predicción dispone de los dos vecinos superiores. Por lo tanto, el uso de WPP requiere una interconexión entre los distintos procesadores para obtener los datos de las dos CTU superiores, no que no resulta de ningún modo problemático en los dispositivos actuales que hacen uso de varios procesadores y ejecuciones multi-hilo.

4.10 Software de Referencia

4.10.1 Modelo de test HM

JCT-VT estableció el modelo de test HM16 en una serie de reuniones mantenidas en Sapporo del 30 de junio al 7 de julio de 2014 y ha sido completado con diversas actualizaciones en sucesivas reuniones. El documento *High Efficiency Vídeo Coding (HEVC) Encoder Description v16* (*HM16*) [2] proporciona una descripción del software HM16, en concreto una breve explicación de las diferentes herramientas de codificación que contiene, cómo el codificador las utiliza y de las opciones de configuración existentes. Este software permite la evaluación de las innovaciones tecnológicas incluidas en el estándar HEVC, ofreciendo un software de referencia en el que realizar modificaciones para evaluar la viabilidad de los algoritmos propuestos. Es ampliamente utilizado en la literatura existente como caso base, permitiendo obtener unos resultados que posibilitan la realización de una comparativa con el software sin ninguna modificación y respecto a otros trabajos previos. Se trata de un software sin optimizar que no explota el paralelismo a nivel de instrucción (*Instruction Level Parallelism, ILP)* en arquitecturas de tipo *Very Long Instruction Word (VLIW)*. El software HM tampoco está preparado para ejecuciones multi-hilo (*multithreading*) ni para codificación de varios planos en paralelo (*frame threading*). Los tiempos de codificación empleando este software son muy elevados, siendo totalmente inviable su ejecución en tiempo real. A pesar de ello, como se ha comentado previamente, este software es utilizado como referencia en la mayoría de publicaciones existentes por lo que será empleado para la obtención de resultados y posterior comparación con el estado del arte. HM implementa todas las técnicas descritas en el estándar HEVC, siendo utilizadas aplicando el método de "fuerza bruta", por lo que se ejecutarán todas las combinaciones existentes. De esta forma, el software HM establece la cota máxima de compresión de vídeo posible con la menor pérdida de calidad visual asociada.

4.10.1.1 Configuraciones preestablecidas

HM soporta tres tipos de estructuras de predicción que son utilizadas en las condiciones de test comunes establecidas en [3] y que son empleadas para realizar los experimentos de los algoritmos propuestos y su comparación con el estado del arte. Estas estructuras son conocidas como *intra-only*, *low-delay* y *random access*. Cada una de estas estructuras se encuentra definida en un fichero de configuración que se proporciona con la distribución de HM.

En la configuración intra-only (también denominada *all intra*), cada imagen que conforma la secuencia de vídeo es codificada como tipo I, por lo tanto, cada imagen puede ser codificada/descodificada sin hacer uso de imágenes pertenecientes a otros instantes de tiempo. Empleando esta configuración, el orden de codificación/descodificación coincide con el orden de presentación.

En el caso de la configuración low-delay, existen dos variaciones: low-delay P y low-delay B. En estas configuraciones, únicamente se codifica la primera imagen como tipo I. El resto de imágenes que componen la secuencia de vídeo son codificadas como slices inter tipo P en la configuración low-delay P o como slices inter tipo B, para la configuración low-delay B. En ambos casos, sólo se puede utilizar como referencia a imágenes que precedan en orden de presentación a la que se esté codificando. En la Figura 4.24, se muestra una representación gráfica del orden de presentación utilizando esta configuración.

Las flechas indican las imágenes de referencia que se emplean en la codificación/descodificación de cada imagen y los números, el orden de codificación. Al igual que en el caso de la configuración intra-only, el orden de codificación/descodificación coincide con el orden de presentación. En la Figura 4.24 también se pueden apreciar los valores de QP asociados a cada imagen y que son dependientes del valor empleado en la imagen tipo I (*QPI*). El valor QPI es indicado como parámetro de entrada al codificador.

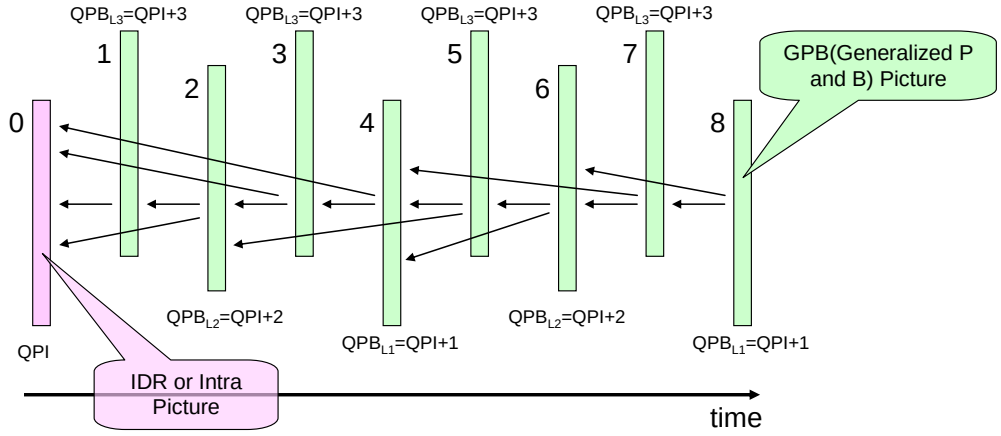


Figura 4.24. Configuración low-delay.
Imagen extraída de [2].

La configuración random-access utiliza una estructura jerárquica de codificación de imágenes tipo B. En la Figura 4.25 se muestra una representación gráfica del orden de presentación de los distintos tipos de imagen empleando la configuración random-access. Al igual que en la Figura 4.24, los números empleados en la Figura 4.25 indican el orden de codificación y las flechas, las imágenes que son usadas como referencias.
Aproximadamente cada segundo se codifica una imagen tipo I teniendo en cuenta la tasa de imágenes por segundo del formato de vídeo de entrada a la codificación. En este caso, todas las imágenes tipo inter son codificadas como tipo B, aunque se distingue entre las que pueden ser usadas como referencia y las que no. En la Figura 4.25, también se muestran los valores de QP asociados a cada imagen, valores dependientes del parámetro de entrada QPI.

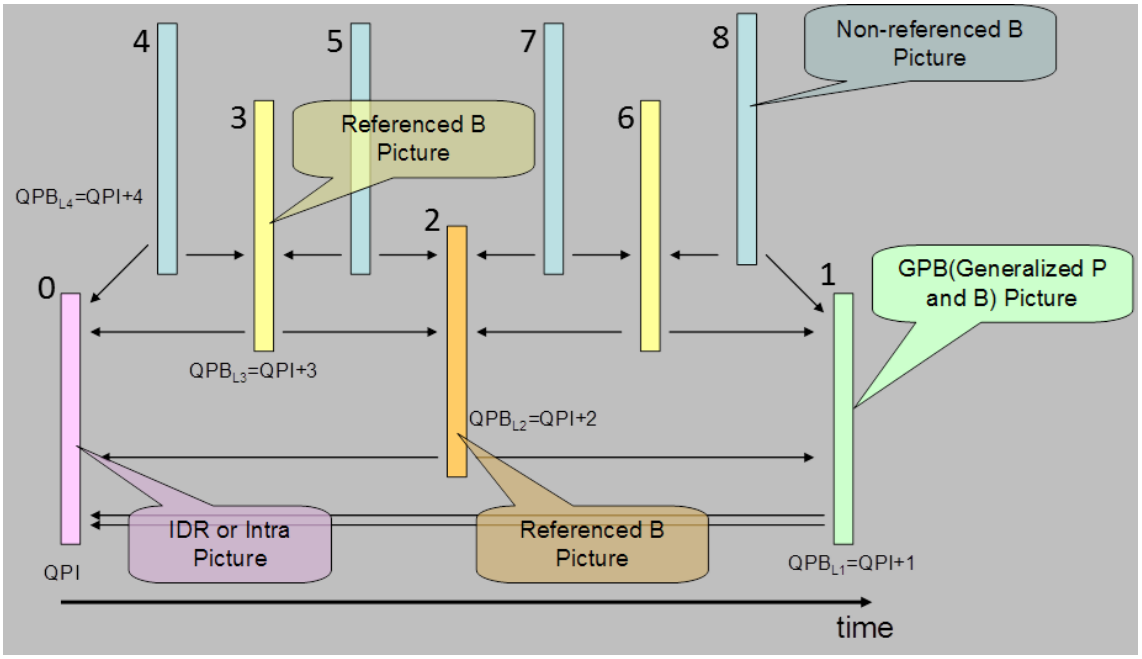


Figura 4.25. Configuración random-access.
Imagen extraída de [2].

4.10.1.2 Flujo de codificación intra

HM ha adoptado un algoritmo simplificado para la predicción intra basado en 4 estadios. En el primer estadio denominado *Rough Mode Decision (RMD),* se realiza el cómputo del coste asociado a los 35 modos de predicción intra disponibles utilizando como función de coste SATD. El coste asociado a cada modo de predicción es calculado utilizando la Ecuación 4.5.

$$Cost = SATD\,(Residue) + \lambda * Mode\_bits. \qquad (4.5)$$

Donde SATD, es la distorsión asociada al residuo tras aplicar el modo de predicción intra que se evalúa. λ, es el multiplicador lagrangiano (realmente, sqrt(λ)). Mode_bits, son los bits utilizados para indicar el modo de predicción en el bitstream. Son obtenidos a partir del uso de un pseudo-CABAC que utiliza unas tablas pre-calculadas para acelerar el proceso.

A la salida del estadio RMD se obtiene un conjunto de N modos de predicción candidatos. El número N depende del tamaño de PU, tal y como se indica en la Tabla 4.3.

Tabla 4.3. Número de candidatos (N) a la salida de RMD en función del tamaño de PU.

Tamaño PU	Nº de candidatos (N)
4x4	8
8x8	8
16x16	3
32x32	3
64x64	3

En el segundo estadio, se realiza una comprobación de si los modos más probables de predicción (*Most Probable Modes, MPM*) están ya incluidos. Si los MPM no están

incluidos se añaden al conjunto de candidatos, pudiéndose añadir hasta tres modos más. Los MPM son obtenidos a partir de bloques vecinos mediante un proceso estandarizado que conforma una tabla. Cuando uno de los MPM es seleccionado como modo de predicción intra ganador, el codificador únicamente envía al equipo descodificador un índice a dicha tabla, con el correspondiente ahorro en bits que esto supone.

En la tercera fase, RDO es aplicado al conjunto de modos candidatos conformado por la salida de RMD y los MPM, para obtener un modo ganador. Sobre este modo ganador se aplica otro proceso recursivo RDO para proceder a dividir el CB en diferentes TB en busca de obtener el tamaño óptimo de TB. El proceso de división de un CB en TB es conocido como *residual quadtree (RQT).*

4.10.1.3 Flujo de codificación inter

En la Figura 4.26 se muestra un diagrama del algoritmo de decisión de modo de predicción inter. A la salida de este proceso se obtendrá el mejor modo de predicción encontrado para una determinada CU, pudiendo ser *MODE_SKIP* o *MODE_INTER*. Si se escoge MODE_SKIP, no se permite continuar la sub-división en PU más pequeñas y los parámetros que describen el movimiento son asignados a la CU, indicando como tamaño de PU el modo de partición 2Nx2N. Utilizando MODE_SKIP los parámetros de movimiento no son codificados y para su obtención en el descodificador se utiliza el proceso de derivación que se aplica al modo merge comentado en el apartado 4.4.1. En el caso de MODE_INTER, hasta ocho modos de partición pueden ser escogidos, tal y como se indicó en la Figura 4.3. A nivel de PU, la predicción inter puede utilizar el modo merge o la predicción puramente inter en la que se indican los parámetros de movimiento de una forma explícita.

Como se puede observar en la Figura 4.26, el primer modo que se evalúa es el modo de partición más grande, 2Nx2N. El flujo que se presenta se ejecuta para cada depth, comenzando con depth=0. Si el parámetro *CTUMaxSize* del codificador HM es igual a 64, entonces depth=0 equivale al tamaño 64x64 (N=32). En el flujo de decisión aparecen las condiciones de parada prematura *EarlySkip* [4], *CBF_Fast* [5] y *Early_CU* [2]. EarlySkip evalúa si el vector de movimiento asociado a MODE_INTER aplicando el tipo de partición 2Nx2N es (0, 0) y si la codificación de este modo contiene coeficientes transformados distintos de 0. Si ambas condiciones se cumplen, se indica que el mejor modo hasta el momento es MODE_SKIP y se evalúa la condición Early_CU. Cuando no se ejecuta la parada prematura EarlySkip, se procede a evaluar la condición CBF_Fast. CBF_Fast comprueba si MODE_INTER aplicando el tipo de partición 2Nx2N contiene coeficientes transformados distintos de 0. Si se cumple la condición, se procede a evaluar la condición Early_CU pero previamente indicando como mejor modo, MODE_INTER PART_2Nx2N. Si no se cumplen las condiciones estipuladas por EarlySkip y CBF_Fast, se procede a evaluar MODE_SKIP aplicando una función de coste J_{mode} para la predicción asociada a los distintos vectores de movimiento candidatos derivados del proceso indicado para el modo merge. La Ecuación 4.6, define J_{mode}.

$$J_{mode} = (SSE_{luma} + w_{chroma} * SSE_{chroma}) + \lambda_{mode} * B_{mode}, \qquad (4.6)$$
$$w_{chroma} = 2^{(QP - QP_{chroma})/3}.$$

Siendo SSE, la métrica de distorsión empleada para la luminancia y la crominancia de la predicción realizada respecto al bloque de entrada a la codificación. λ_{mode} es el lagrangiano

definido en [2] cuyo valor depende del valor de QP utilizado, número de planos tipo B empleados en la codificación de la secuencia, si la slice que se está codificando es tipo P o B, y la configuración de codificación empleada (low-delay o random-access). B_{mode} indica el coste en bits asociada a la codificación de un determinado modo de predicción.

A continuación, se evalúa MODE_INTER para el tamaño de partición NxN cuando se alcanza el máximo valor de depth permitido siempre y cuando no se trate de N=4, ya que las PU 4x4 están deshabilitadas por defecto para slices tipo inter mediante el flag *inter_4x4_enabled*. Posteriormente, se evalúa si este modo contiene coeficientes distintos de cero (condición CBF_Fast). Si la condición se cumple se evalúa la condición Early_CU pero estableciendo como mejor modo hasta el momento MODE_INTER PART_NxN. La condición CBF_Fast se comprueba tras la evaluación de cada modo que aparece en la Figura 4.26, pero no se ha incluido en el flujo del programa por simplicidad.
En las siguientes fases de la comparación se evalúan consecutivamente los modos de partición Nx2N y 2NxN. Tras la evaluación de cada modo se comprueba si para ese modo se cumple la condición CBF_Fast, si se cumple, se procede a evaluar la condición Early_CU estableciendo previamente como mejor modo hasta el momento MODE_INTER PART_Nx2N o 2NxN según el estadio en el que nos encontremos. Antes de ser evaluados los modos AMP son comprobadas una serie de condiciones para decidir si se procede a su evaluación o no. Dichas comprobaciones quedan resumidas en la Tabla 4.4. Las condiciones dependen del mejor modo de partición anterior a la evaluación de los modos AMP y de los modos de predicción y de partición asociados al valor anterior de depth que contiene la PU que se evalúa en el proceso de división recursivo (conocido como *parent CU*).

Tabla 4.4. Condiciones para la evaluación de los modos AMP.

Condición	**Acción**
El mejor modo de partición es 2NxN	Se evalúa 2NxnU y 2NxnD
El mejor modo de partición es Nx2N	Se evalúa nLx2N y nRx2N
El mejor modo de partición es 2Nx2N y no es ni skip ni merge	Se evalúan todos los modos AMP
El modo de partición de parent CU es AMP	Se evalúa únicamente el modo de predicción merge para todos los modos de partición AMP
El modo de partición de parent CU es 2Nx2N y la evaluación de parent CU se realizó (no se produjo alguna condición que llevase a saltarse su evaluación)	Se evalúa únicamente el modo de predicción merge para todos los modos de partición AMP
parent CU es intra y el mejor modo de partición es 2NxN	Se evalúa únicamente el modo de predicción merge para los modos de partición 2NxnU y 2NxnD
parent CU es intra y el mejor modo de partición es Nx2N	Se evalúa únicamente el modo de predicción merge para los modos de partición nLx2N y nRx2N
El tamaño de la CU que se evalúa es 64x64	No se evalúa ningún modo AMP

Para cada modo AMP se comprueba si se cumple la condición CBF_Fast, si se cumple, se procede a evaluar la condición Early_CU estableciendo previamente como mejor modo el modo AMP que corresponda. Tras realizar las evaluaciones de las predicciones inter se procede a evaluar la posibilidad de realizar una predicción intra para el modo de partición 2Nx2N. Si depth es igual al valor máximo que puede tomar, entonces se evalúa el modo de predicción intra asociado a las cuatro particiones NxN que componen el bloque 2Nx2N.

Para cada evaluación realizada, si el coste obtenido por ese modo es menor que en las evaluaciones anteriores, se fija este nuevo coste como el mínimo y el modo como ganador. Después de la evaluación de la predicción intra se actualiza el valor de B_{mode} teniendo en cuenta el coste en bits asociado al modo ganador (la indicación del modo, vectores de movimiento e índice de referencia, vector de movimiento predicho, modo de predicción intra, etc.) y a la codificación del residuo. Finalmente se evalúa la condición Early_CU, si el mejor modo escogido es MODE_SKIP entonces se detiene el proceso de división recursivo y se procesa la siguiente CTU. En caso contrario, y siempre que no estemos en el máximo valor de depth, se procede al siguiente nivel de división recursiva incrementando el valor actual de depth. Si el valor de depth es el máximo, se procede a evaluar la siguiente CTU.

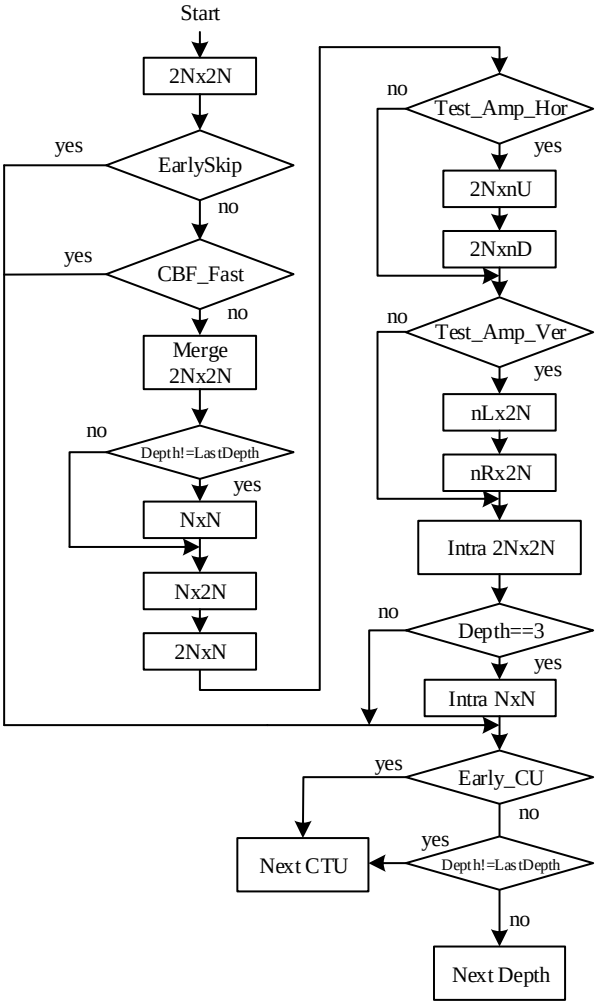


Figura 4.26. Flujo de ejecución asociado al algoritmo de decisión de modo de predicción inter.

4.10.2 Codificador en tiempo real x265

La realización de pruebas en un entorno que permita la codificación en tiempo real es uno de los objetivos que se persiguen en la presente Tesis. Al no posibilitar el software HM el establecimiento de un entorno de este tipo, hubo que buscar una alternativa viable. x265 es un proyecto de software libre que implementa un codificador HEVC optimizado para su ejecución en tiempo real [6]. En función de la calidad que se desee obtener ofrece diversos presets de configuración, cuanto mayor calidad se obtenga mayores tiempos de ejecución se requerirán, por lo que en función de la aplicación a la que se destine su uso habrá que llegar a un determinado compromiso. x265 está implementado para ser ejecutado en procesadores multi-núcleo, haciendo uso de ejecuciones multi-hilo para soportar el paralelismo que aporta la herramienta WPP y la codificación de varias imágenes al mismo tiempo. Se trata de un software optimizado a nivel de instrucción que incluye código para explotar ILP en función de la arquitectura hardware del microprocesador en el que se ejecute (SSE2-4, AVX, AVX2, etc.).

x265 permite codificar de una manera eficiente vídeos de resolución HD en tiempo real en procesadores de un coste medio. El código es proporcionado bajo los términos de una licencia de tipo GNU GPLv2 (*General Public License version 2*).

El flujo de ejecución que utiliza x625 es similar al de HM, por lo que la integración de los algoritmos desarrollados sobre HM puede considerarse como una tarea abordable, aunque no puede llevarse a cabo de una forma directa.

4.11 Nuevos estándares

4.11.1 Inconvenientes asociados a HEVC

El coste monetario del uso de HEVC no es despreciable debido principalmente a que en el diseño de las herramientas que incorpora HEVC han colaborado además de JCVT-VC empresas como Ericsson, Panasonic, Qualcomm, Sharp y Sony que reclaman regalías de uso mediante ciertas entidades denominadas *license pools*: MPEG LA, HEVC Advance y Velos Media [7]. El hecho de que el sistema de licencias no esté centralizado supone un inconveniente, ya que el pago de regalías puede llegar a convertirse en un proceso demasiado engorroso. A este problema, se añade el hecho de que existan ciertas compañías no asociadas a ninguna de las tres entidades que también exigen pago por sus licencias incluidas en el HEVC.

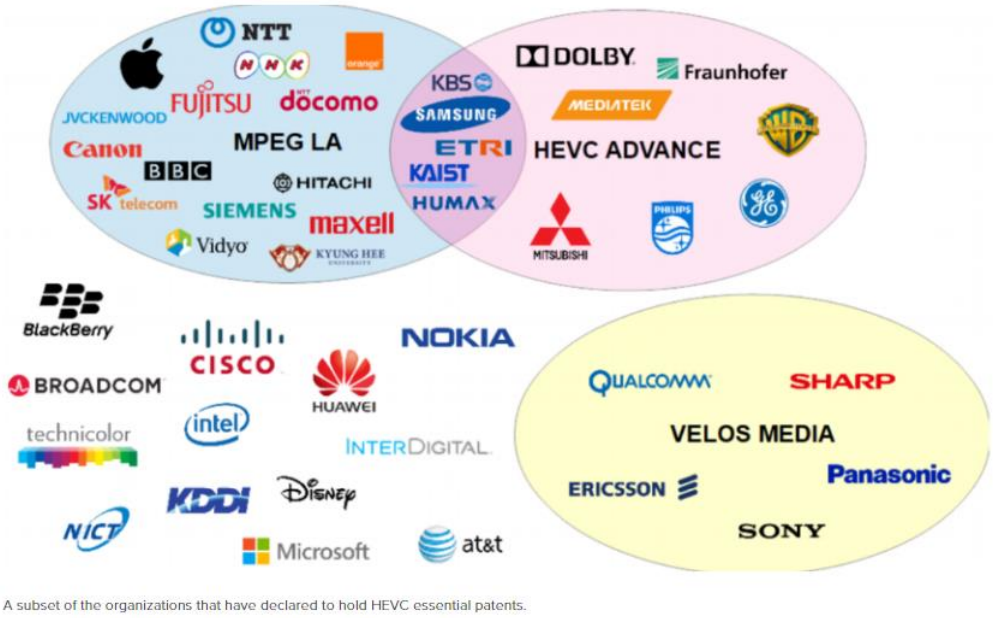


Figura 4.27. Subconjunto de organizaciones que poseen patentes sobre herramientas del HEVC.
Imagen extraída de [8].

Debido a estos inconvenientes, han surgido otros estándares libres de regalías como VP9 o AV1 para competir con HEVC, lo que ha llevado a que las entidades asociadas a HEVC eliminen ciertas regalías para intentar combatir dicha competencia. Respecto a esto, en marzo de 2018, HEVC Advance anunció que dentro de las tasas por uso de licencias, se eliminaban las asociadas a la distribución y uso de contenidos que no utilicen un soporte físico. Este hecho ha sido un gran avance para la adopción de HEVC por parte de las plataformas de distribución de contenidos como Netflix.

Para poder implementar un codificador o descodificador basado en el estándar HEVC MPEG-LA establece las siguientes regalías [9]:

- Para empresas que distribuyan de 0-100000 unidades/año, exención en el pago de tasas.

- Para empresas que distribuyan más de 100000 unidades/año, 0.20$/unidad una vez superada la cantidad de 100000 unidades.
- Se fija una cantidad máxima a pagar por una determinada empresa en 25 millones de dólares por año.

Como se puede observar, para determinadas empresas el coste asociado a la comercialización de productos basados en HEVC puede llegar a ser considerable. Los fabricantes de teléfonos móviles, dispositivos con hardware embebido capaz de procesar HEVC, tabletas gráficas, etc. superan ampliamente las 100000 unidades por año. Aun así, estos costes asociados no han sido impedimento para que en 2017 Apple anunciase soporte nativo de HEVC en todos sus dispositivos, o para que Intel y Nvidia incluyan en la mayoría de sus GPU módulos hardware dedicados exclusivamente a la codificación/descodificación de HEVC.

4.11.2 Alternativas a HEVC

En este apartado se describirán las alternativas existentes al uso de HEVC. Se trata de estándares de codificación de vídeo que son contemporáneos a HEVC o que han sido o van a ser finalizados con posterioridad.

4.11.2.1 VP9

Es un estándar de compresión de vídeo *open-source* y *royalty-free* desarrollado por Google. Estas dos propiedades son las dos grandes virtudes que ofrece este estándar respecto al HEVC. La primera versión se liberó en diciembre de 2012, y su principal objetivo es reducir la tasa de datos generada un 50% respecto a su predecesor VP8.
En términos de calidad y tamaño de bitstream los resultados que ofrece HEVC y VP9 dependen considerablemente del tipo de secuencia utilizada, aunque estudios realizados por Netflix reflejan una superioridad del 20% de una implementación HEVC respecto a VP9 en términos de eficiencia en la compresión [10]. Estos datos no son concluyentes ya otros estudios modificando la configuración empleada por los codificadores han demostrado que ofrecen prestaciones muy similares [11].

La principal ventaja que ofrece VP9 respecto a HEVC es su integración en la mayoría de navegadores WEB (Google Chrome, Mozilla Firefox, Microsoft Edge, Android, Chromium y Opera) frente a HEVC que sólo se encuentra implementado en el navegador de Android, Safari y Microsoft Edge (únicamente si el dispositivo donde se ejecuta dispone de descodificación por hardware) [12]. VP9 está incluido en el formato multimedia WebM orientado a ser utilizado en HTML5. Entre otros, WebM sobre HTML5 es utilizado por Youtube (propiedad de Google) para la distribución de sus contenidos. Adobe Flash Player no soporta VP9, siendo éste el principal problema que ha sufrido VP9 en su adopción. Por otro lado, la mayor parte de migraciones de Flash a HTML5 en aplicaciones WEB fueron realizadas cuando VP9 no se encontraba en un estado maduro de implementación.

Respecto a su situación actual respecto a HEVC, VP9 está siendo utilizado más ampliamente que HEVC en entornos WEB, mientras que HEVC ocupa el resto del mercado.

4.11.2.2 AOMedia Vídeo 1 (AV1)

AV1 [13] ha sido desarrollado por *Alliance for Open Media (AOMedia)*, alianza que surgió como respuesta a la situación de las gestiones de licencias de HEVC. Se considera el sucesor de VP9, ya que está ampliamente basado en el trabajo realizado por Google sobre dicho códec. AV1 además comprende los esfuerzos realizados por Mozilla en el Proyecto Daala [14] y por Cisco en Thor [15]. El consorcio AOMedia está constituido por las empresas antes mencionadas Google, Mozilla y Cisco además de Netflix y Amazon, junto a algunos fabricantes de dispositivos hardware como AMD, ARM, Intel y Nvidia.

AV1 es un estándar de vídeo, ya que AOMedia publicó en junio del 2018 la especificación 1.0. Uno de los principales alicientes para el uso de AV1 es que cumple las especificaciones para ser utilizado en *Internet Vídeo Codec (NETVC)*, un proyecto del organismo de estandarización *Internet Engineering Task Force (IETF).* IETF tiene como objetivo la redacción de documentación técnica de alta calidad que influencie en la forma en la que la gente [16] diseña, usa y administra Internet.

AV1 es un estándar de reciente aparición, no habiendo todavía codificadores en tiempo real que se encuentren lo suficientemente optimizados para permitir su ejecución en plataformas hardware de bajo/medio coste.

4.11.2.3 Codificadores basados en contenido y contexto

Bajo este apartado no se presentará ningún estándar si no una familia de codificadores que pretenden mejorar la compresión teniendo en cuenta el contenido de un vídeo y/o el contexto asociado a la codificación, aplicando técnicas de *computer vision y machine learning*. En la literatura existente al respecto, son denominados *Context-Content Aware Encoding (CAE)* [17].

El concepto de contenido se refiere a las características presentes en el vídeo en cuestión. Por ejemplo, una secuencia compuesta por un busto parlante (una entrevista o una videoconferencia) no presenta el mismo movimiento que una secuencia de deporte. En el primer caso, no sería necesario el uso de una amplia zona de búsqueda en la estimación de movimiento por lo que se podría reducir la ventana de búsqueda y se podría destinar dicho gasto computacional a algoritmos para mejorar la calidad visual de la cara, puesto que será la parte de la imagen donde la persona que visualiza el vídeo prestará mayor atención.

El contexto asociado a la codificación depende de gran cantidad de parámetros y factores. Entre los más interesantes cabe resaltar la arquitectura hardware del dispositivo donde se ejecuta la codificación, para tenerla en cuenta a la hora de tomar una serie de decisiones para reducir el consumo, o ajustar los parámetros del codificador en función de la capacidad de cómputo del dispositivo en cuestión.

4.11.2.4 MPEG-VVC (Versatile Video Coding)

Versatile Video Coding (VVC) es la evolución del estándar HEVC. Tras la finalización de HEVC las investigaciones han continuado evolucionando. Algunas de las técnicas que no fueron incluidas en HEVC han seguido siendo evaluadas, refinadas y mejoradas; al igual que se han desarrollado estudios relevantes sobre nuevas técnicas.

En octubre de 2015 se creó *JVET (Joint Video Exploration Team)* una nueva colaboración entre ITU-T VCEG e ISO/IEC MPEG con el objetivo de crear un grupo de investigación para definir un nuevo estándar que mejore las prestaciones tanto de HEVC

como AV1, reduciendo la tasa en un 50% respecto al HEVC para una misma calidad subjetiva. La denominación de versátil que incluye el nombre de este nuevo estándar se refiere a que pretende cubrir las necesidades referentes a alto/bajo bitrate, grandes/pequeñas resoluciones, *High Dynamic Range (HDR),* codificación omnidireccional 360º, etc.

En marzo de 2017 se produjo un evento determinante, denominado *Call For Evidence (CFE)* que constituye el proceso mediante el cual MPEG solicita evidencias tangibles que demuestren que una nueva tecnología mejora las prestaciones de las anteriores y justifique el esfuerzo a realizar para crear un nuevo estándar. Se creó un software para explorar estas propuestas denominado *Joint Video Exploration Model (JEM)* basado en el software de referencia de HEVC (HM). Los resultados obtenidos en su primera versión han sido muy positivos teniendo en cuenta que existen muchas técnicas que todavía no han sido implementadas, siendo cercanos a una reducción del 34% en bitrate para una misma PSNR respecto a HEVC [18].

En octubre del 2017 se produjo el evento *Call For Proposal (CFP)*, evento en el que se solicita a los expertos del sector que propongan nuevas técnicas para ser incluidas en el estándar. Se recibieron propuestas desde 33 organizaciones, consiguiendo una de reducción superior al 40% en el tamaño de bitstream respecto al HEVC. La primera versión de modelo de test de VVC denominado *VVC Test Model (VTM)* está siendo elaborado por *Fraunhofer Heinrich Hertz Institut*e [19], al igual que se sucedió en el caso de H.264 y HEVC.

Entre las herramientas estudiadas que ofrecen mejores resultados se encuentra el uso del tamaño de CTU de 128x128 y el uso de múltiples tipos de particiones no soportados en HEVC, como la subdivisión en tres particiones. Para decidir el tipo de partición a utilizar durante la codificación de manera prematura y así reducir los tiempos de ejecución, serán de gran ayuda los algoritmos basados en clasificación de texturas como algunos de los diseñados e implementados en la presente Tesis.

Mientras que el tiempo de ejecución de la codificación en el software de referencia de AV1 es aproximadamente un 2000% superior a HEVC, el tiempo de ejecución en VVC es cercano a 4 veces el de HEVC [20], lo que posibilitará su implementación en dispositivos existentes.

VVC actualmente está siendo desarrollado pero como se ha comentado, los primeros resultados son muy prometedores. En su planificación temporal se encuentra definida una fecha límite para la creación de la primera versión del estándar en octubre de 2020.

4.12 Referencias

[1] Lainema J., Bossen F., Han W., Min J., Ugur K. (2012). Intra Coding of the HEVC Standard. IEEE Transactions on Circuits and Systems for Video Technology, 22(12).

[2] McCann K., Rosewarne C., Bross B., Naccari M., Sharman K., Sullivan G. (2014). High efficiency video coding (HEVC) encoder description 0v16 (HM16). JCT-VC High Efficiency Video Coding N14 703.

[3] Bossen F. (2012). Common test conditions and software reference configurations. JCT-VC Document, JCTVC-K1100.

[4] Yang J., Kim J., Won K., Lee H., Jeon B. (2011). Early SKIP detection for HEVC. JCT-VC Document, JCTVC-G543.

[5] Choi K., Jang E. S. (2011). Coding tree pruning based CU early termination. JCT-VC Document, JCTVC-F092.

[6] x265 Project. (Junio de 2019). Recuperado de http://x265.org/

[7] IBC Congress. (Enero de 2019). Recuperado de https://www.ibc.org/delivery/codec-wars-the-battle-between-hevc-and-av1/2710.article

[8] Samuelsson J. (2018). Codec wars. IBC 2018 Congress. Amsterdam.

[9] MPEGLA. (Enero de 2019). Recuperado de http://www.mpegla.com/main/programs/hevc/documents/hevcweb.pdf

[10] Ozer, J. (Enero de 2019). Streaming Media. Netflix Finds x265 20% More Efficient than VP9. Recuperado de http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/Netflix-Finds-x265-20-More Efficient-than-VP9-113346.aspx

[11] Ozer, J. (Mayo de 2018). Streaming Learning Center. HEVC: Rating the contenders. Recuperado de https://streaminglearningcenter.com/wp-content/uploads/2017/05/Comparing_Best_HEVC_Codec-1.pdf

[12] Wikipedia. (Enero de 2019). HTML5 video. Recuperado de https://en.wikipedia.org/wiki/HTML5_video#Browser_support

[13] Mozilla Research. (Enero de 2019). AV1 & Media Codecs. Recuperado de https://research.mozilla.org/av1-media-codecs/

[14] Xiph.org. (Enero de 2019). Daala Video Compression. Recuperado de https://xiph.org/daala/

[15] Cisco. (Enero de 2019). World, Meet Thor – a Project to Hammer Out a Royalty Free Video Codec. Recuperado de https://blogs.cisco.com/collaboration/world-meet-thor-a-project-to-hammer-out-a-royalty-free-video-codec

[16] Internet Engineering Task Force (IETF). (Junio de 2019). Recuperado de https://www6.ietf.org/

[17] Siglin, T. (Enero de 2019). Streaming Media. Buyers' Guide to Content- and Context-Aware Encoding 2018. Recuperado de http://www.streamingmediaglobal.com/Articles/Editorial/Featured-Articles/Buyers-Guide-to-Content--and-Context-Aware-Encoding-2018-123407.aspx

[18] Bross B. (Junio 2018). Versatile Video Coding (VVC). The emerging new standard. Recuperado de https://www.mcg.upv.es/wp-content/uploads/2018/06/IEEE_BMSB_2018_Keynote_Day2_BenjaminBross_Public.pdf

[19] Fraunhofer Heinrich Hertz Institute. (Enero de 2019). Versatile Video Coding (VVC). Recuperado de https://jvet.hhi.fraunhofer.de/

[20] Feldmann C. (Enero de 2019). Bitmovin. VVC Video Codec – The Next Generation Codec. Recuperado de https://bitmovin.com/vvc-video-codec/?utm_campaign=Newsletter&utm_medium=email&_hsenc=p2ANqtz-9eA6b6J6U4n9YSMssZdpOqRfATU6tR1R-SI_MoXwMa8Uy-g7sZ0ds8WrQ7ImdczcK3zKpa7o42D590AYvrMB6yxfYpgA&_hsmi=68737465&utm_content=68737464&utm_source=hs_email&hsCtaTracking=aab78af9-169b-42ad-9acf-85132eda840a%7C4be493ce-b8c6-4483-b142-d22544a19bee

Capítulo 5

Contribuciones. Codificación HEVC.

5.1 Introducción

En el presente capítulo se mostrarán una serie de optimizaciones que se han llevado a cabo con el objetivo de reducir la carga computacional asociada a la codificación HEVC sin un aumento considerable en la tasa de datos generada y con una pérdida de calidad visual imperceptible. Los algoritmos de simplificación están basados en un análisis realizado sobre la imagen de entrada sin procesar, con el objetivo de encontrar regiones de la imagen que contengan una serie de características que permitan tomar un conjunto de decisiones prematuras.

El hecho de trabajar con la imagen de entrada directamente permite que el análisis pueda realizarse a nivel de plano completo o en el propio bucle de codificación de CTU en el cuerpo del codificador. En la implementación realizada se ha optado por trabajar a nivel de plano para poder realizar fácilmente el análisis de la imagen en un co-procesador, eliminando las esperas innecesarias simplemente añadiendo el tiempo de una trama de vídeo (plano) al retardo total del sistema codificador. De esta forma, el co-procesador dispone del tiempo de una trama de vídeo para realizar el pre-análisis antes de que el procesador a cargo de realizar la codificación requiera los resultados obtenidos.

5.2 Estado del Arte

Dentro de las diferentes ideas existentes en la literatura para mitigar el alto coste computacional asociado a la codificación HEVC se incluye la detección de regiones con texturas homogéneas. Está ampliamente difundida la idea de que la división de bloques de la imagen con texturas homogéneas en bloques de tamaños inferiores no aporta una considerable mejora en la reducción del coste RDO asociado [1-8] y por lo tanto, en la reducción de la tasa de datos generada, en relación al enorme gasto computacional que conlleva dicho proceso de división recursivo.

El mismo concepto puede ser aplicado a las zonas de la imagen en las que existe una dirección espacial predominante en la distribución de los píxeles que la componen [1, 9-10]. Para la detección de este tipo de regiones la gran mayoría de publicaciones utilizan algoritmos basados en gradientes [2-3, 8, 11-12] en los que se asocian conjuntos de valores de gradientes a determinados modos de predicción. El uso de gradientes también se emplea para la clasificación de las regiones con texturas homogéneas. En ambos casos, los algoritmos se basan en la utilización de máscaras de 3x3 píxeles y requieren el cálculo tanto de la amplitud como de la arcotangente asociada al ángulo del gradiente, además del algoritmo propiamente dicho de clasificación, por lo que son procesos computacionalmente costosos. Si no existe una determinada dirección predominante se clasifican como texturas homogéneas, lo cual puede conducir a decisiones erróneas puesto que una distribución caótica de los píxeles tampoco presenta una dirección que predomine.

En [13] los autores utilizan la desviación media absoluta (*mean absolute deviation*) para la detección de texturas homogéneas, otros trabajos se basan en una detección utilizando la varianza [4-5, 11] o la media de diferencias absolutas (*mean of absolute differences*) entre píxeles trabajando conjuntamente con la desviación estándar de la varianza [7]. Sin embargo, todas estas aproximaciones al problema son computacionalmente costosas.

La correlación entre la CU a codificar, los estadios anteriores del proceso de división recursivo y los vecinos espaciales ha sido estudiada por Shen et al. [14], trabajo en el que se propone un algoritmo para simplificar la decisión del tamaño de bloque a aplicar y el modo de predicción intra a utilizar basándose en el tamaño y modos escogidos en decisiones anteriores (estadios previos y vecinos espaciales). El uso de los vecinos espaciales añade dificultad al procesado en paralelo debido a las dependencias existentes, que pueden solventarse mediante la implementación de un sistema que evalúe correctamente la disponibilidad de dichos vecinos en función de su posición dentro de la imagen y de que haya finalizado su procesado. Debido a la introducción de las esperas asociadas a la finalización del procesado de CU vecinas, siempre se verá mermada la ventaja que ofrece el uso de las herramientas asociadas a HEVC para aplicar paralelismo, principalmente WPP, así como la utilización de otros hilos de ejecución en paralelo para la estimación de movimiento y/o predicción intra. Aun presentando esta clara desventaja, la correlación existente entre la CU a codificar y los vecinos espaciales y temporales, ha sido utilizada en otros trabajos [13, 15-17]. En ninguno de estos trabajos se aborda el problema relativo al cómputo paralelo, puesto que los algoritmos han sido implementados únicamente utilizando el software HM que no dispone de la posibilidad de aplicar paralelismo.

Otras aproximaciones al problema de la optimización de un codificador HEVC se basan en la utilización de la función de coste RD [15-16, 18] y/o Hadamard [18] junto a la definición de umbrales que determinan la toma de una serie de decisiones. En [18] el número de modos de predicción intra candidatos es reducido empleando el coste asociado

a la transformada Hadamard para la evaluación entre los distintos modos, e incorpora una serie de paradas prematuras basadas en el coste RD de la CU que se evalúa y de las sub-CU tras aplicar la división. El principal problema de este tipo de soluciones es el hecho de que la obtención del coste RD es una operación con una exigente demanda computacional. Los tiempos de ejecución del software HM son muy elevados por lo que se podría llegar a enmascarar el coste computacional asociado a un algoritmo de simplificación cuya implementación en un codificador destinado a ser ejecutado en tiempo real sea totalmente inabordable. Dicho de otro modo, en implementaciones en tiempo real, podría ser más costoso el análisis que requiere el algoritmo de simplificación que la propia reducción de tiempo de ejecución que logra en el codificador de vídeo.

En [19] se aplica un método estadístico basado en el estudio de las probabilidades de que la división de un bloque suceda, con el objetivo de simplificar la decisión de tamaño de bloque a aplicar. En general, los algoritmos basados en estadísticas ofrecen excelentes resultados en media, pero para algunas secuencias que presenten características que varían continuamente durante el vídeo (p. ej. secuencias que contienen muchos cambios de escena) las prestaciones se ven claramente disminuidas, incluso pudiendo llevar a decisiones erróneas debido a que no todos los vídeos presentan las mismas características ni distribuciones de probabilidad.

Diversos parámetros espaciales y temporales son utilizados en [20] por medio del análisis utilizado en el filtrado SAO, pero dicho filtrado es habitualmente deshabilitado en codificadores de bajo coste que se ejecutan en tiempo real debido a las exigencias computacionales de dicho filtrado.

La detección de regiones estáticas dentro de la imagen o de regiones dentro de las cuales todos los bloques presentan un movimiento homogéneo ha sido utilizada en diferentes estudios [21-23] para llevar a cabo diversas simplificaciones en la codificación. Tanto dentro de la región estática como en el caso del área con movimiento homogéneo, todos los bloques que componen la zona de la imagen presentan la misma información de movimiento que en el caso concreto de la región estática, tendrá un valor que refleje un movimiento nulo o muy pequeño. Por el contrario, el movimiento homogéneo puede corresponderse con vectores de movimiento muy elevados, simplemente define que todos los bloques presentan un movimiento muy similar. En ambos casos, la utilización de un algoritmo adaptativo para determinar el tipo de partición a emplear no es necesario para describir de una manera precisa el movimiento. La detección de dichas categorías implica el uso de una estimación de movimiento [21-22, 24-27], proceso que es computacionalmente muy costoso [28-29]. Debido a la naturaleza intensiva que presentan los algoritmos de búsqueda de movimiento, son procesos idóneos para ser implementados en dispositivos operando como co-procesadores, tales como GPU y/o FPGA. La idea de aplicar procesos de codificación simplificados en regiones temporalmente homogéneas ya fue utilizada para el estándar H.264 [21-22, 24-26]. En [21] se utiliza una estimación de movimiento a nivel de 16x16 píxeles para obtener el residuo de un bloque y decidir posteriormente si el macrobloque es temporalmente homogéneo o no. En el caso de macrobloques no homogéneos, se ejecuta una estimación de movimiento extra a nivel de bloques de 8x8 píxeles y se compara con el coste anteriormente calculado a nivel de 16x16 para decidir qué modo de predicción inter finalmente utilizar. El operador de gradiente de Sobel es utilizado en [22] sobre un conjunto de vectores de movimiento obtenidos mediante una estimación de movimiento a nivel de bloques 4x4 para generar un mapa de gradientes que describen el movimiento de la imagen y realizar una clasificación posterior. Los trabajos presentados en [25] y [26] utilizan la desviación media de los vectores de movimiento de los vecinos espaciales y temporales para la detección de regiones temporalmente homogéneas. En lo que respecta al estándar HEVC, en [27] se presenta un

análisis en busca de regiones temporalmente homogéneas pero se ignora por completo el movimiento existente en interior de la propia CU a codificar, centrándose para realizar la clasificación en el movimiento existente en CU vecinas.

Además de los inconvenientes asociados a algunos de estos trabajos que ya se han comentado, la mayoría de algoritmos presentan flujos condicionales complejos difíciles de implementar en arquitecturas basadas en GPU o FPGA. En todas las publicaciones comentadas, los algoritmos han sido implementados únicamente en el software de referencia HM y por lo tanto, sin tener en cuenta un sistema de codificación en tiempo real ni el uso de herramientas para aplicar cómputo en paralelo.

5.3 Optimizaciones basadas en homogeneidad espacial

En este apartado se describirán una serie optimizaciones basadas en el análisis de la imagen de entrada en busca de regiones que contengan un cierto grado de homogeneidad espacial. Este tipo de áreas de la imagen, debido a las características que presentan son susceptibles de explotar de una forma muy eficiente tanto la redundancia espacial como la temporal.

En los algoritmos que se han desarrollado, los bloques que presentan homogeneidad espacial son clasificados en dos categorías en función de si presentan texturas homogéneas o una clara direccionalidad espacial en la distribución de los píxeles que los componen.

5.3.1 Detección de texturas homogéneas

5.3.1.1 Análisis de la imagen

En una fase inicial se evaluaron diversos métodos existentes en la literatura para la detección de zonas con texturas homogéneas dentro de la imagen. Se descartaron directamente todos aquellos que requieran de información referente a la codificación de CU previas, puesto que se desea trabajar con la imagen de entrada directamente en un proceso independiente al de codificación. De esta forma, se facilitará la implementación del análisis en un co-procesador si se desea, permitirá su portabilidad entre distintas plataformas HW y posibilita que el análisis pueda ser utilizado en otros estándares de codificación posteriores al HEVC.

El uso de la relación existente entre las componentes de baja frecuencia (DC) y media-alta (AC) permite realizar una detección muy eficiente de este tipo de regiones como se demuestra en [30]. Un bloque en el que sólo exista componente DC presenta una textura totalmente plana, por lo que resulta evidente que cuanto mayor sea la predominancia de la componente DC respecto a las componentes AC, mayor homogeneidad presentará el bloque que se esté tratando. En la Figura 5.1.a se muestran las diferentes texturas obtenidas dependiendo del número de coeficientes existentes. En [30], los autores utilizan esta relación no con el objetivo de reducir la complejidad de la codificación, sino para descartar de una estimación de movimiento perceptual las regiones con texturas homogéneas por los problemas que les plantean en su algoritmo.

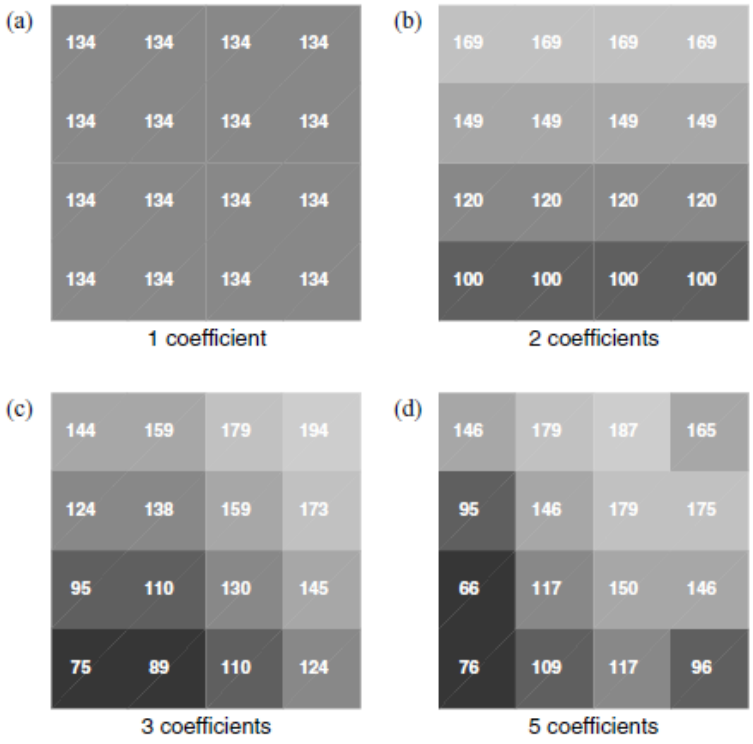


Figura 5.1. Bloque 4x4 reconstruido en función del número de coeficientes utilizado. El único coeficiente que aparece en (a) es el DC. Imagen extraída de [31].

El principal problema que presenta el uso de esta relación es que requiere trabajar en el dominio de la frecuencia, lo que conlleva un considerable gasto computacional [1], puesto que habría que aplicar una DCT a toda la imagen a analizar. Para solventar este problema se aplica la Relación de Parseval [32]. De acuerdo a la Relación de Parseval, los dominios de la frecuencia y el tiempo son representaciones equivalentes de la misma señal y deben contener la misma energía. Por lo tanto, es posible trabajar en el dominio temporal a nivel de píxel calculando previamente la energía. En la Ecuación 5.1 se define el cálculo del ratio DC (DC_R).

$$DC_R = \frac{\text{DC Energy}}{\text{DC Energy+AC Energy}} \overset{\text{Parseval's Relation}}{=} \frac{\left(\sum_{i,j}^{N} pixel(i,j)\right)^2 \frac{1}{N^2}}{\sum_{i,j}^{N} pixel(i,j)^2}. \tag{5.1}$$

Donde pixel (i, j) corresponde a los diferentes píxeles que componen la imagen dentro de un bloque de tamaño NxN. La energía DC es calculada como la media de los valores de los píxeles contenidos en el bloque. DC Energy + AC Energy es la energía total del bloque. Si el valor de DC_R supera un determinado umbral (DC_{TH}) entonces, el bloque será clasificado como espacialmente homogéneo. Una interesante ventaja de trabajar con el ratio es que permite la utilización de un mismo umbral independientemente del tamaño del bloque. La determinación del umbral a utilizar es un aspecto crucial para que la aplicación del algoritmo no afecte a las prestaciones del codificador. Por lo tanto, resulta fundamental alcanzar una solución de compromiso adecuada en términos de pérdida de calidad visual,

incremento de la tasa de datos generada y reducción de tiempos de ejecución respecto a un caso base del codificador sin las modificaciones incluidas. La primera aproximación consistió en la determinación de un valor de umbral inicial (DC_{TH1}), analizando los valores DC_R asociados a una solución existente. Para tal propósito se utilizó el algoritmo presentado en [13], el cual está basado en la desviación media absoluta, empleando diversas secuencias de vídeo con características muy dispares. Después de realizar este primer ajuste, se visualizaron todos los resultados comprobando que la clasificación realizada por el algoritmo basado en DC_{TH1} no se correspondía en gran número de casos con lo que el HVS clasificaría como un área de texturas homogéneas. Por ello, se realizaron diversas pruebas de ensayo y error incrementando levemente el valor de DC_{TH1} hasta alcanzar una clasificación más cercana a la obtenida por el ojo humano. El umbral obtenido empíricamente tras estas modificaciones fue denominado DC_{TH2}. En la Figura 5.2, se muestra una comparativa entre el resultado obtenido por el algoritmo presentado en [13] y por el algoritmo basado en DC_{TH2}. Las zonas en verde oscuro muestran las diferentes zonas clasificadas como espacialmente homogéneas para el primer plano de la secuencia *BasketballDrill* [33].

Como se puede observar en la Figura 5.2 comparando ambas soluciones, la clasificación basada en ratio DC se aproxima más a lo que el HVS considera que posee texturas homogéneas. En la Figura 5.2.b se clasifican erróneamente como texturas homogéneas las líneas pintadas en el suelo y algunas irregularidades en el parqué, mientras que en la Figura 5.2.c dichos defectos se corrigen. El algoritmo basado en el ratio DC es más restrictivo, lo cual permitirá obtener una relación de compromiso más eficiente en términos de incremento de tasa de datos generada y reducción en tiempos de ejecución, tal y como se muestra en las comparativas de prestaciones entre diferentes trabajos que son presentadas en apartados posteriores.

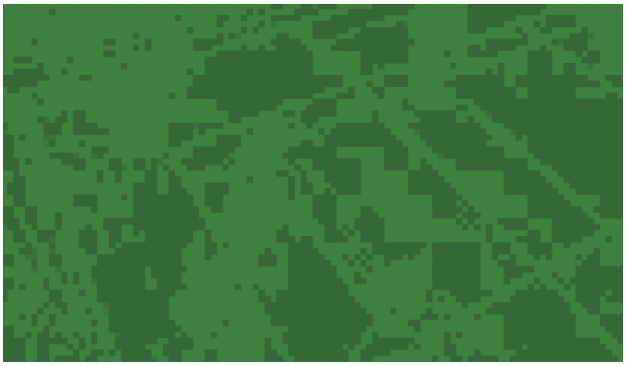
La primera optimización que se plantea aplicar sobre el flujo del codificador HEVC es detener el proceso recursivo de división de la imagen en los bloques clasificados como espacialmente homogéneos. Por ello, se estableció un nuevo proceso de refinamiento del umbral a aplicar analizando únicamente los valores de ratio DC obtenidos en bloques que poseen texturas homogéneas (clasificados así utilizando DC_{TH2}), en los que el coste RD asociado a la utilización de bloques grandes es inferior o muy similar al coste RD después de aplicarle al bloque en cuestión la división recursiva. En la Figura 5.3 se muestran los resultados obtenidos para el tamaño de bloque 32x32 en la secuencia *blue_sky*, utilizando como codificador el software HM v16.2 y un valor de QP = 30.

Después de estudiar minuciosamente los resultados, es posible refinar el valor de DC_{TH2} para mejorar el algoritmo de clasificación. Como se puede apreciar en la Figura 5.3, un valor de umbral igual a 0.9996 es un buen candidato para este caso. Siguiendo un procedimiento similar con diferentes tipos de secuencias y configuraciones (todos los tamaños de CU posibles, valores de QP, etc.) finalmente se obtuvo el valor óptimo del umbral DC_{TH}.

En la Figura 5.4 se puede apreciar el resultado obtenido tras aplicar el algoritmo de clasificación en las secuencias *aspen* y *BQTerrace*. Las regiones que poseen texturas homogéneas se resaltan en color gris claro. Como resultado de la aplicación del algoritmo de clasificación se obtiene un mapa binario para cada tamaño de bloque posible (64x64, 32x32, 16x16, 8x8 y 4x4), donde cada posición indica si el bloque presenta texturas homogéneas o no.

a)

b)



c)

Figura 5.2. Clasificación de texturas homogéneas.
a) Imagen Original. b) Clasificación basada en desviación media absoluta.
c) Clasificación basada en el ratio DC empleando DC_{TH2}.

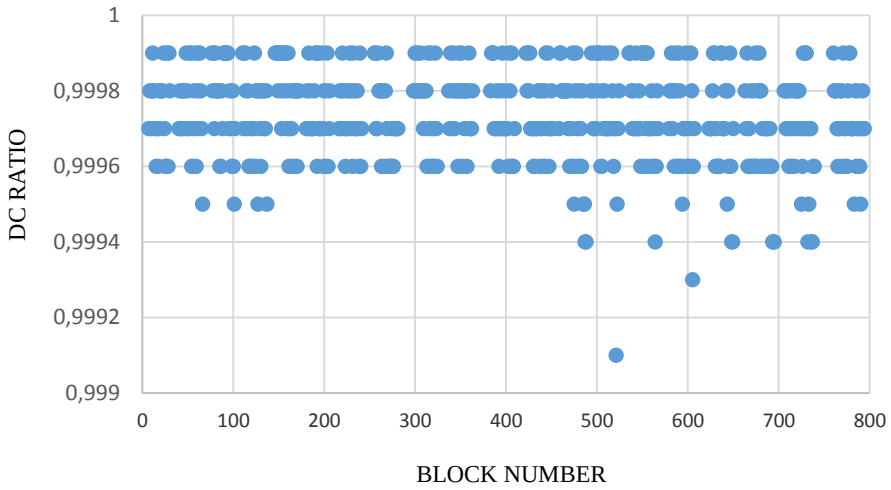
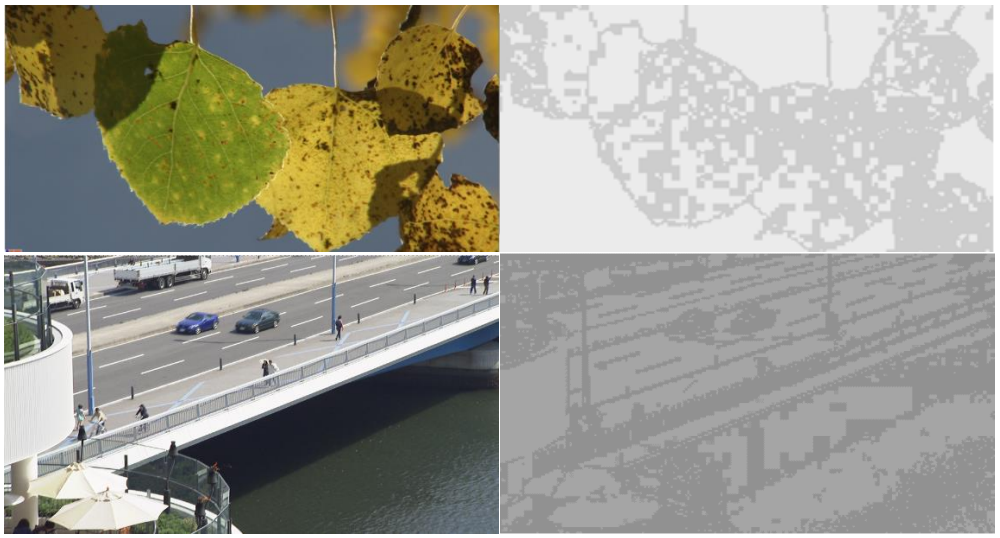


Figura 5.3. Ratios DC obtenidos para bloques con texturas homogéneas en la secuencia blue_sky.



a) b)

Figura 5.4. Resultado de la clasificación de texturas propuesta.
a) Imagen original extraída de las secuencias aspen y BQTerrace.
b) Clasificación de regiones con texturas homogéneas (gris claro).

5.3.1.2 Implementación de la clasificación de texturas homogéneas

El algoritmo de clasificación fue inicialmente implementado en lenguaje C en el bucle de codificación de CTU en el software HM v16.2. Gracias a que la clasificación se realiza trabajando únicamente con la imagen de entrada y a la independencia del algoritmo respecto al valor de QP utilizado, los parámetros de codificación y de las decisiones tomadas en CU vecinas, es posible implementarla fácilmente en un co-procesador.

Debido a que tanto el software HM como x265 se ejecutan en microprocesadores, se optó para la implementación del algoritmo de clasificación por una GPU trabajando como co-procesador. La principal causa de la elección de este dispositivo es la rapidez y

simplicidad con la que se puede crear un sistema funcional formado por un host y un co-procesador.

La implementación fue realizada inicialmente en CUDA, el lenguaje propio de las GPU de Nvidia [34], y finalmente en OpenCL [35]. OpenCL es un lenguaje de programación para cómputo paralelo libre de regalías que puede ser ejecutado en diversas plataformas (microprocesadores, GPU, FPGA, etc.) mediante el uso de los *drivers* correspondientes. Por lo tanto, OpenCL proporciona una interoperabilidad que permite utilizar GPU de distintos fabricantes, e incluso habilita la posibilidad de utilizar una FPGA empleando la misma porción de código. Este factor fue determinante a la hora de finalmente decidir implementar en OpenCL los algoritmos que realizan el análisis de la imagen de entrada, ya que uno de los objetivos autoimpuestos, es que los algoritmos que se diseñen puedan ser implementados en diversas plataformas hardware en un futuro. Otros trabajos presentan complicaciones para implementar sus algoritmos en co-procesadores debido a la utilización de QP [7, 10, 36] o de información relativa a las CU vecinas [14, 15, 16], lo que implica que deban de implementarse obligatoriamente dentro del bucle de codificación de CTU. Por lo tanto, su implementación exige incluir un complejo sistema de esperas que merma considerablemente las prestaciones del sistema.

Profundizando en la implementación del algoritmo de clasificación, el cálculo del ratio DC se realiza inicialmente sobre toda la imagen a nivel de bloque 4x4. Este cálculo se realiza mediante el correspondiente kernel de la GPU del que se obtiene el mapa binario asociado a este tamaño. Para no replicar cálculos, los resultados de las operaciones que se realizan sobre los píxeles de entrada en el primer kernel son reutilizados por el kernel posterior para crear el mapa binario a nivel de bloque 8x8. El resultado para el tamaño 8x8 es reutilizado para los cálculos a nivel 16x16, el de 16x6 para el tamaño 32x32 y finalmente el de este último para el cálculo de bloques 64x64.

Para la implementación del kernel que calcula propiamente el ratio DC a nivel 4x4 se hace uso de la memoria interna de la propia GPU. Cada procesador de la GPU es encargado de trabajar con una fila de cuatro píxeles siguiendo el siguiente flujo:

1. Carga en memoria interna de los cuatro píxeles de coordenadas (x, y), (x+1, y), (x+2, y) y (x+3, y).
2. Se realiza la suma de los cuatro píxeles y el resultado se almacena en un array situado en memoria interna (*sharedSumPixels*).
3. Se eleva al cuadrado cada píxel y se realiza la suma de los cuatro cuadrados. El resultado se almacena en memoria interna (*sharedSumPixelsSQ*).
4. Si el procesador está trabajando con la última fila de un bloque 4x4 (y==3), tras esperar mediante una barrera (*barrier*) a que otros procesadores que estén trabajando con líneas anteriores del mismo bloque 4x4 finalicen, se realiza la suma de las 4 filas de píxeles utilizando cuatro posiciones del array *sharedSumPixels* y almacenando el resultado a nivel de bloque 4x4 en una posición del array *SumPixels*. De manera equivalente, utilizando el array *sharedSumPixelsSQ* se realiza la suma de todos los píxeles al cuadrado del bloque 4x4 y se almacena en el array *SumPixelsSQ.*
5. Utilizando los valores almacenados en los arrays *SumPixels* y *SumPixelsSQ*, se procede al cálculo a nivel de bloque 4x4 de la energía DC y de la energía total del bloque.
6. Se obtiene el DC ratio para el bloque 4x4 dividiendo las energías conforme la Ecuación 5.1 y se compara con el umbral DC_{TH}. Si se supera el valor de DC_{TH}, se indica en el mapa a nivel 4x4 que el bloque ha sido clasificado como espacialmente homogéneo. El mapa está situado en memoria externa ya que almacenará la

clasificación a nivel 4x4 de toda la imagen. Para reducir la memoria externa empleada y los movimientos de memoria entre microprocesador y GPU, realmente se utiliza un único array para almacenar todos los mapas binarios asociados a los distintos tamaños de bloque. En el bit 0 se sitúa el resultado de la clasificación para el tamaño 4x4, en el bit 1 para el tamaño de bloque 8x8, y así sucesivamente hasta almacenar en el bit 4 el resultado para el tamaño de bloque 64x64.

Como se ha comentado, para el cálculo del ratio DC a nivel de bloque 8x8 se hace uso de los arrays *SumPixelsSQ y SumPixels* a nivel de 4x4 ya que sumando las cuatro posiciones de estos arrays que incluyen los píxeles de un bloque 8x8, se obtendrá la suma a nivel 8x8 de todos los píxeles y la suma de los píxeles al cuadrado. A su vez, la suma de los 4 bloques 4x4 es almacenada en los arrays *SumPixelsSQ8x8 y SumPixels8x8* a nivel de 8x8, que son reutilizados en el cálculo asociado al tamaño 16x16. El resultado de 16x16 se almacena en *SumPixelsSQ16x16 y SumPixels16x16*, arrays que son utilizados para el tamaño 32x32. De manera equivalente, el resultado de 32x32 se almacena en *SumPixelsSQ32x32 y SumPixels32x32*, arrays que son reutilizados para el tamaño 64x64. Como la energía de cada bloque depende de su tamaño, es calculada independientemente para cada tamaño de bloque en el kernel asociado a dicho tamaño. La suma de los píxeles es utilizada para el cálculo de la media que permite saber el valor de la energía DC, y la suma del cuadrado de los píxeles para la energía total del bloque (AC + DC). Para los tamaños 8x8, 16x16 y 32x32, se utiliza un mismo kernel de OpenCL variando los parámetros de entrada/salida. Respecto a éstos, el tamaño 64x64 presenta la única peculiaridad de no almacenar el resultado de la suma de los píxeles ni de la suma del cuadrado de los píxeles en ningún array, ya que es el tamaño de bloque más grande permitido por HEVC. En la Figura 5.5 se ilustra gráficamente el proceso seguido para la obtención de SumPixels32x32. De manera análoga se obtiene SumPixelsSQ32x32, pero lógicamente trabajando con el cuadrado de los píxeles.

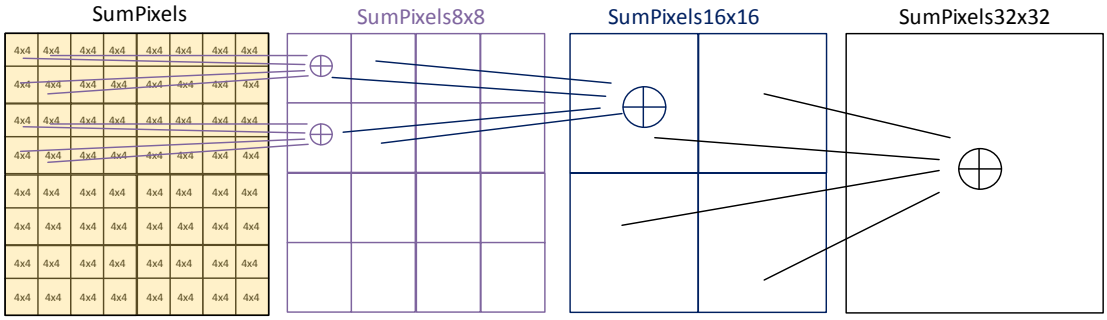


Figura 5.5. Proceso de reutilización de cálculos en el algoritmo de clasificación de texturas.

En la Figura 5.6 se muestran los movimientos de memoria existentes entre los distintos kernels y entre GPU y microprocesador. La imagen y el mapa binario resultado de la clasificación de texturas deben de ser utilizados tanto por el microprocesador como por el co-procesador. El microprocesador será el encargado de realizar la codificación propiamente dicha, tarea para la cual requiere de la imagen de entrada. La GPU analiza las texturas presentes en la imagen por lo que también debe de estar disponible para su acceso. Los búferes SumPixels y SumPixelsSQ asociados a cada tamaño sólo deben de ser accedidos por la GPU para la reutilización de datos entre las distintas ejecuciones de los kernels. Finalmente, el mapa binario resultado de la clasificación realizada en el co-

procesador debe de utilizarse en el procesador para llevar a cabo las optimizaciones correspondientes en el proceso de codificación.

En el caso de usar diferentes espacios de memoria entre el procesador y co-procesador, se requerirá realizar copias de la imagen y del mapa binario resultado. Actualmente, existen diversas plataformas hardware que permiten compartir espacios de memoria entre procesador y co-procesador, como es el caso de las plataformas de Intel con GPU embebida. En el caso de este fabricante, es posible compartir espacios de memoria entre microprocesador y GPU bajo ciertas restricciones, eliminando de esta forma los movimientos de memoria entre ellos. En [37] se describe con mucha claridad las restricciones en la reserva de memoria que se deben cumplir y el proceso a seguir para posibilitar el uso de un espacio común de memoria, proceso conocido como *OpenCL Zero Copy*.

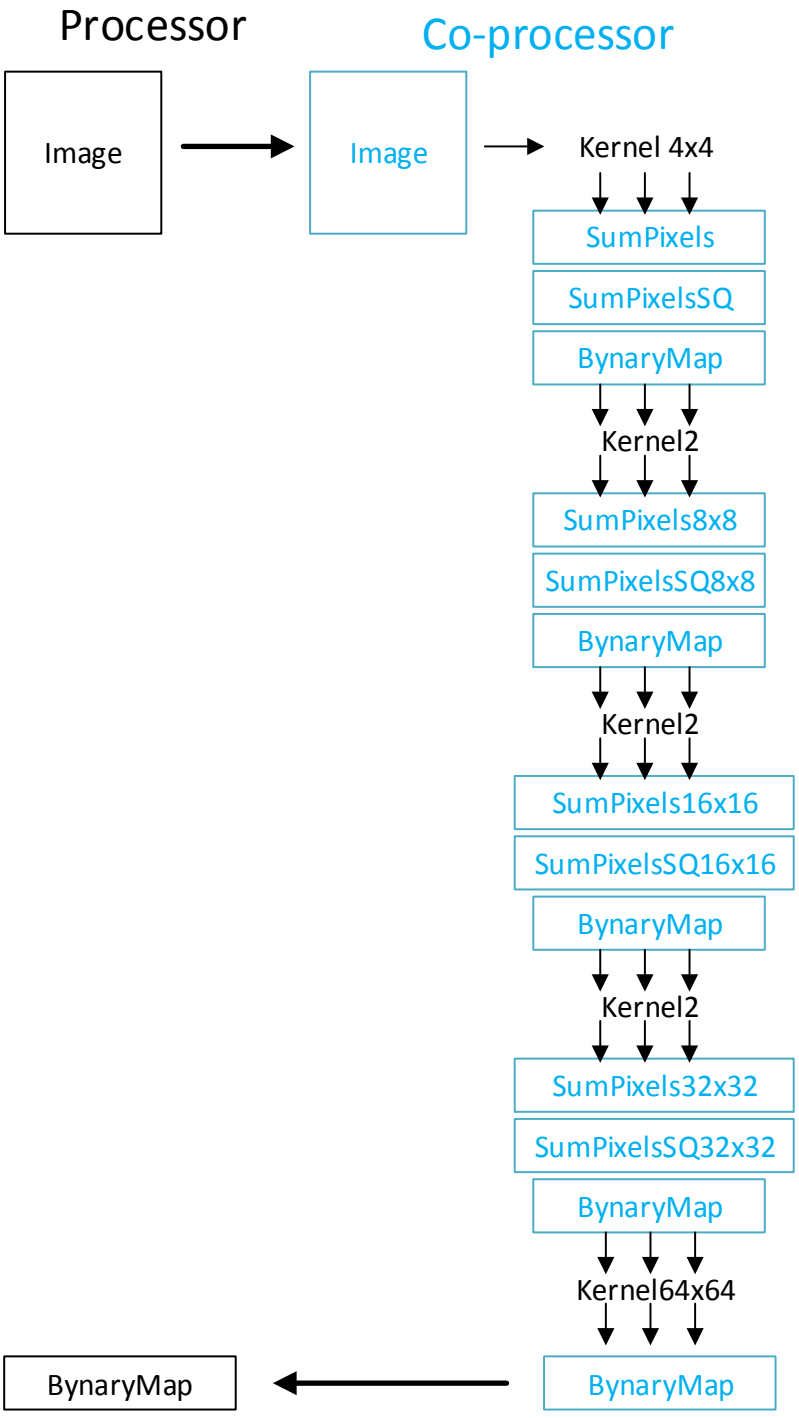


Figura 5.6. Requerimientos de memoria en el algoritmo de clasificación de texturas.

5.3.1.3 Algoritmo de selección de tamaño de bloque

Como primera aproximación al problema de la optimización de un codificador HEVC, en este apartado se expone la optimización del algoritmo de decisión de tamaño bloque haciendo uso del mapa binario obtenido en la clasificación basada en texturas.

5.3.1.3.1 Parada Prematura

Dentro del bucle de codificación de CTU del codificador, una vez obtenido el coste de un determinado tamaño de bloque, se consulta el mapa binario asociado a dicho tamaño. Si el bloque fue clasificado como espacialmente homogéneo por presentar homogeneidad en sus texturas, entonces se detiene el proceso recursivo, procediendo a evaluar la siguiente CU dentro de la CTU actual, ya que la división de la CU no aportará mejora en la eficiencia de la codificación. En el caso de que la CU que se esté evaluando sea la última CU de la CTU actual, entonces se procede a procesar la primera CU de la siguiente CTU. Esta simplificación es realizada tanto en CU tipo intra como de tipo inter. Es importante resaltar que el mapa binario a nivel de 4x4 no es utilizado en el algoritmo de selección de tamaño de bloque, al ser éste el tamaño inferior que contempla el estándar HEVC.

5.3.1.3.2 Optimización tamaño de CTU

Un importante factor que afecta enormemente a los tiempos de ejecución del codificador HEVC es el tamaño de la CTU que se emplee. Este tamaño definirá a su vez, el máximo tamaño de CU que se puede aplicar. El valor por defecto de tamaño de CTU en el software HM v16.2 (parámetro $CTU_{MaxSize}$) es 64x64. La utilización de tamaños de CTU menores reduce el número de estadios a evaluar dentro del proceso recursivo de división, reduciendo considerablemente la carga computacional del codificador. Sin embargo, la reducción del tamaño de CTU produce un aumento de la métrica BD-Rate que indica una pérdida de prestaciones del codificador tal y como se muestra en la Tabla 5.1. En la Tabla 5.1 se muestran los resultados obtenidos utilizando un tamaño de CTU igual a 32x32 ($CTU_{MaxSize} = 32$) respecto a la configuración por defecto donde $CTU_{MaxSize} = 64$. Al tratarse de un parámetro de configuración del software de referencia HM asociado al estándar HEVC, para la realización de las pruebas se emplearon exclusivamente las condiciones de test recomendadas por JCT-VT en [33].

Tabla 5.1. Prestaciones del codificador utilizando CTUMaxSize = 32.

Sequences	Intra-Only		Low-Delay P		Random-Access	
	BD-Rate	ΔTime (%)	BD- Rate	ΔTime (%)	BD- Rate	ΔTime (%)
Class A	0.7%	-10.17	3.7%	-15.73	2.3%	-15.90
Class B	0.6%	-8.27	5.0%	-14.15	3.3%	-12.53
Class C	0.3%	-9.15	2.2%	-11.76	1.1%	-12.17
Class D	0.2%	-8.27	1.6%	-9.66	0.8%	-9.97
Class E	0.7%	-9.70	9.0%	-12.20	4.0%	-12.65

El incremento en el valor de BD-Rate es bastante bajo para la configuración intra-only, pero empeora considerablemente para las configuraciones que hacen uso de predicción inter (low-delay P y random-access), donde este valor puede alcanzar hasta un 9%, tal y como se indica en la Tabla 5.1. Con el objetivo de mantener la reducción de carga computacional que supone impedir la utilización de CTU de tamaño 64x64 y de reducir el incremento de BD-Rate asociado, se decidió estudiar el número de bloques de tamaño 64x64 que serían categorizados como espacialmente homogéneos por el algoritmo de clasificación de texturas homogéneas propuesto. En tal análisis se concluyó que para la configuración low-delay P el 27% de bloques 64x64 en secuencias 1080p (*Class B*, en la Tabla 5.1) y el 36 % en secuencias 720p (*Class E*) presentaban texturas homogéneas. El valor obtenido para las secuencias que suponen mayor incremento de BD-Rate podía alcanzar hasta el 49% (secuencia *Basketball Drive*) o 55.8% (secuencia *Johnny*). Por lo tanto, habilitar el uso de bloques 64x64 únicamente en regiones clasificadas como

espacialmente homogéneas debería suponer una reducción de carga computacional sin afectar considerablemente a la calidad visual y a la tasa de compresión de datos obtenida. En la propuesta realizada, el tamaño de CTU se fija a 64x64, pero sólo se permite el uso de CU de tamaño 64x64 para los bloques con texturas homogéneas. Como en este tipo de bloques no se permite la sub-división de la CU, su uso implica tanto una reducción en el tiempo de codificación como una mejora en la eficiencia de la codificación tal y como se muestra en la Tabla 5.2. Los resultados que muestran en la Tabla 5.2 incluyen la optimización realizada en el algoritmo de selección de tamaño de bloque. Por lo tanto, sólo se utiliza el tamaño 64x64 en bloques con texturas homogéneas y se detiene el proceso recursivo en los bloques que presentan texturas homogéneas, tanto en 64x64 como en 32x32, 16x16 u 8x8. Si se compara la Tabla 5.1 con la Tabla 5.2, se puede observar como se ha reducido considerablemente el incremento de BD-Rate y se ha aumentado la reducción en el tiempo de codificación empleado.

Tabla 5.2. Prestaciones del codificador utilizando bloques 64x64 únicamente en regiones con texturas homogéneas y deteniendo el proceso de división recursivo en bloques clasificados como homogéneos.

	Intra-Only		Low-Delay P		Random-Access	
Sequences	BD-Rate	ΔTime (%)	BD- Rate	ΔTime (%)	BD- Rate	ΔTime (%)
Class A	0.4%	-31.33	0.4%	-12.31	0.5%	-11.92
Class B	0.8%	-33.33	0.6%	-17.49	0.8%	-18.39
Class C	0.5%	-23.99	0.3%	-7.34	0.5%	-7.18
Class D	0.3%	-23.20	0.3%	-6.58	0.3%	-6.99
Class E	0.9%	-43.56	1%	-28.48	0.8%	-29.17

5.3.1.4 Selección del modo de predicción intra

El algoritmo de decisión del modo de predicción intra a utilizar es computacionalmente muy costoso por lo que se trata un proceso sujeto a optimización. El objetivo de esta sección es la reducción de los modos de predicción intra a evaluar en las regiones que poseen texturas homogéneas. Para tal propósito, se ha realizado un estudio de los modos de predicción intra que son más frecuentemente utilizados en este tipo de regiones. El software utilizado para realizar el estudio fue el codificador HM v16.2 sin ninguna de las modificaciones propuestas. Dicho software evalúa todos los modos de predicción intra posibles empleando RDO, por lo que se puede confiar en que la elección realizada por este software está cercana al ideal en lo referente a reducción de tasa de datos generada para una cota de calidad visual predefinida por el valor de QP utilizado. Para el estudio, se utilizaron secuencias de vídeo que contienen distintos tipos de texturas y con características dispares, empleando todos los valores de QP (0-51) que se permiten en estándar HEVC. En la Figura 5.7 se muestra una gráfica que muestra un histograma de los modos de predicción intra para diferentes valores de QP en la secuencia *pedestrian*.

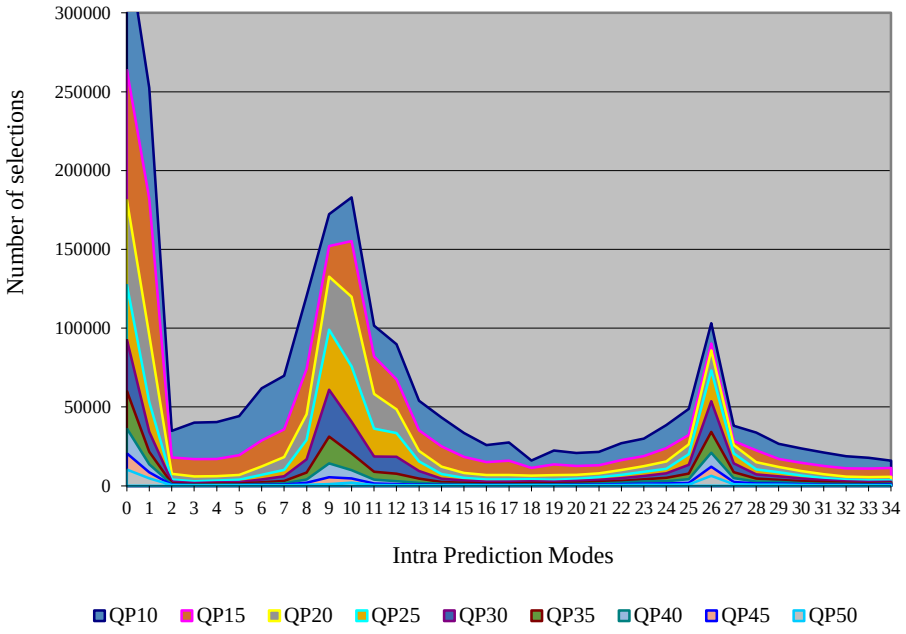


Figura 5.7. Histograma de modos de predicción intra asociados a la secuencia pedestrian utilizando diferentes valores de QP.

Intuitivamente se podría pensar que en las regiones con texturas homogéneas sólo es necesario utilizar los modos DC (modo 1) y Plano (modo 0), ya que los bloques contenidos en este tipo de áreas no presentan ninguna dirección predominante. Analizando el histograma obtenido para distintas secuencias, se observa la necesidad de incluir los modos horizontal (10) y vertical (26) debido a la alta probabilidad que tienen estos modos de ser elegidos como el mejor modo de predicción, tal y como muestran los valores que aparecen en la Figura 5.7 para cualquier valor de QP. Además de estos cuatro modos, se incluyó en la lista de modos candidatos el primer MPM si éste no es uno de los cuatro modos indicados, debido a la alta probabilidad de ocurrencia (34%) que presenta este modo tal y como se indica en [14]. Por lo tanto, un máximo de 5 modos de predicción intra serán evaluados en bloques que presenten texturas homogéneas, siendo éstos: plano, DC, horizontal, vertical y un MPM; si éste no es uno de los anteriores. En [10] los autores utilizan en su algoritmo hasta doce modos de predicción intra. Diez modos son evaluados en [36] comparando costes SATD, a continuación los cinco mejores modos son evaluados mediante RDO. En el trabajo expuesto en [2], se evalúan dieciocho modos para los tamaños 8x8, 16x16 y 32x32, y ocho para el tamaño 4x4. En [12] el número de modos evaluados puede llegar a ser menor, siendo dos, tres o cinco candidatos dependiendo de varios umbrales. Sin embargo, en este estudio sólo el algoritmo de decisión de modo de predicción intra es optimizado, sin tener en cuenta la predicción inter o el algoritmo de decisión de tamaño de bloque.

Debido a que las regiones con texturas homogéneas no presentan complejidad, el número de bits generado presenta un efecto prácticamente despreciable dentro de la función de coste RDO. Por ello, la evaluación basada en RDO ha sido eliminada del algoritmo propuesto para la decisión del modo de predicción intra a aplicar, utilizando para la comparación entre los distintos candidatos únicamente costes basados en la medida de distorsión SATD. Esto permite, como se muestra en la sección de resultados obtener una

importante reducción en los tiempos de ejecución sin afectar a la tasa de datos generada ni a la calidad visual.

5.3.1.5 Predicción inter

Si un bloque presenta texturas homogéneas, el proceso recursivo de división de la imagen es detenido y no son evaluados tamaños inferiores de bloque. Además de esta optimización, es posible obtener un decremento de los tiempos de codificación reduciendo en la predicción inter el número de modos de partición a evaluar dentro del tamaño de CU actual para bloques con texturas homogéneas. Para determinar el número de modos a utilizar, se realizaron diversos experimentos sobre el software HM v16.2, sin ninguna de las modificaciones propuestas implementadas. El estudio se realizó de manera similar al realizado para la predicción intra, empleando diversas secuencias y todos los valores de QP permitidos. En la Figura 5.8, se muestra el histograma de los modos de partición inter seleccionados por el software HM v16.2 para las regiones que poseen texturas homogéneas en la secuencia *pedestrian* utilizando diferentes valores de QP.

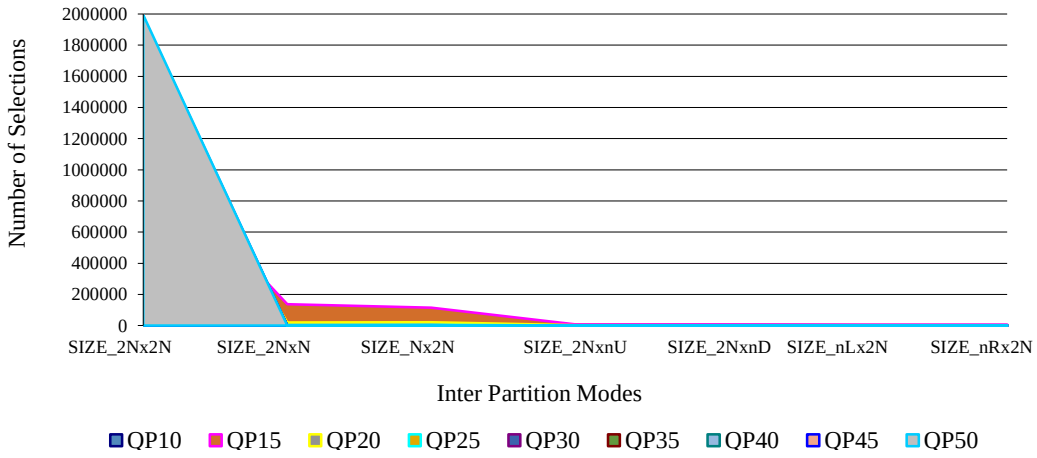


Figura 5.8. Histograma de modos de partición inter asociados a la secuencia pedestrian utilizando diferentes valores de QP.

Como se puede observar en el histograma mostrado en la Figura 5.8, el modo 2Nx2N es indiscutiblemente el modo que ha sido seleccionado mayor número de veces para todos los valores de QP. Para los valores inferiores de QP, también toman importancia los modos 2NxN y Nx2N, viéndose disminuido su uso a medida que el valor de QP aumenta. Se realizó una batería de pruebas exhaustiva utilizando únicamente estos tres modos cuando un bloque presenta texturas homogéneas y se repitieron las mismas pruebas pero reduciendo el número de modos de partición inter que se evalúan a únicamente el modo 2Nx2N. En estas pruebas se observó que la mejora en cuanto a reducción de tasa generada que aportaba la utilización de los modos 2NxN y Nx2N en regiones con texturas homogéneas era despreciable respecto al coste computacional que suponen. En la Tabla 5.3 se muestra un subconjunto de los resultados obtenidos. En dicha tabla se muestra únicamente la media obtenida para todos los valores de QP empleados, con el objetivo de reducir el tamaño de la tabla.

Tabla 5.3. Comparación de prestaciones evaluando los modos de partición inter 2Nx2N, 2NxN y Nx2N vs 2Nx2N.

Sequence	2Nx2N, 2NxN and NxN			2Nx2N			Difference		
	ΔBits (%)	ΔPSNR (dB)	ΔTime (%)	ΔBits (%)	ΔPSNR (dB)	ΔTime (%)	ΔBits (%)	ΔPSNR (dB)	ΔTime (%)
blue_sky 1080p25	-0.10	-0.02	-32.92	0.13	-0.05	-37.02	0.23	-0.03	-4.10
pedestrian 1080p25	0.33	-0.02	-26.04	0.55	-0.04	-29.82	0.22	-0.03	-3.78
riverbed 1080p25	-0.03	-0.01	-4.48	-0.01	-0.01	-5.98	0.02	0.00	-1.50
rush_hour 1080p25	-0.29	-0.02	-30.59	-0.09	-0.03	-34.51	0.20	-0.01	-3.92
parkjoy 1080p25	0.01	0.00	-0.24	0.01	-0.02	-1.64	0.01	-0.02	-1.40
sunflower 1080p25	-0.34	-0.01	-16.50	-0.16	-0.04	-21.30	0.18	-0.03	-4.80
Tractor 1080p25	0.12	-0.02	-7.38	0.15	-0.02	-9.60	0.03	-0.01	-2.22
Stockholm 720p60	-0.43	-0.01	-14.74	-0.32	-0.01	-18.11	0.10	-0.01	-3.37
shields 720p59	-0.01	-0.01	-3.25	0.02	-0.02	-5.01	0.02	-0.01	-1.76

Por lo tanto, en el algoritmo de optimización propuesto, se decidió evaluar únicamente el modo de partición 2Nx2N (que incluye el modo de predicción SKIP), por lo que el número de modos de partición a evaluar se reduce de 8 a únicamente 1, para los bloques que presentan texturas homogéneas.

5.3.1.6 Mejoras subjetivas de la calidad visual

5.3.1.6.1 Modificación del valor de cuantificación

Teniendo en cuenta que el HVS es más sensible a percibir artefactos causados por el proceso de codificación en regiones con texturas homogéneas [38], se decidió aprovechar la clasificación de texturas realizada para mejorar la calidad visual subjetiva por medio de la variación del valor del parámetro de cuantificación (QP) asignado por el algoritmo de control de tasa (RC). Como ya se comentó previamente, valores bajos de QP aumentan la calidad visual de la imagen a expensas de un aumento en el número de bits generado. Por el contrario, un aumento en el valor de QP supone una reducción de calidad visual y una reducción en la tasa de datos generada para esa misma imagen.

Por lo tanto, según el funcionamiento del HVS disminuir el valor de QP en regiones con texturas homogéneas aumentará la calidad subjetiva percibida por el usuario. En una primera aproximación se pensó que la disminución de QP no debería aumentar considerablemente la tasa de datos generada ya que estas zonas no presentan estructuras complejas. Después de realizar una primera batería de pruebas con el RC activo, se observó que a bajos bitrates si se decrementa excesivamente el valor de QP, el aumento del número de bits en regiones con texturas homogéneas al principio de una secuencia de vídeo provoca que el RC reaccione aumentando considerablemente el valor de QP en las imágenes posteriores para conseguir mantener la tasa de bits constante en un segundo. Dicho aumento

de QP, finalmente provoca una pérdida global de la calidad visual percibida. Por ello, se debe determinar un valor de QP que permita aumentar la calidad visual pero sin provocar un aumento excesivo en el número de bits. El leve aumento en el número de bits debería compensarse aumentando el valor de QP en regiones de la imagen que no presenten texturas homogéneas, zonas que permiten un mayor grado de distorsión hasta que el ruido introducido por la codificación sea perceptible [38]. Con vistas a cumplir este objetivo se realizó una nueva batería de pruebas con secuencias que presentan todo tipo de características. Las pruebas se realizaron con el RC desactivado de forma que se aplica un valor de QP constante para toda la secuencia de vídeo, y se aplica en cada prueba un decremento de QP (QP_D) distinto en regiones con texturas homogéneas, buscando determinar el incremento en bits provocado por cada QP_D dado un valor de QP inicial. Por lo tanto, para cada valor de QP se aplican distintos valores de QP_D y se analiza el incremento en el número de bits producido en cada secuencia.

En la Figura 5.9, se muestra una gráfica extraída de dicho análisis en el que se muestra el incremento de tasa de datos generada para diferentes valores de QP, aplicando decrementos de 1 a 7 en las regiones con texturas homogéneas. Los valores mostrados son una media de los valores obtenidos para las secuencias *blue_sky, rush_hour, pedestrian, riverbed, shields* y *stockholm.*

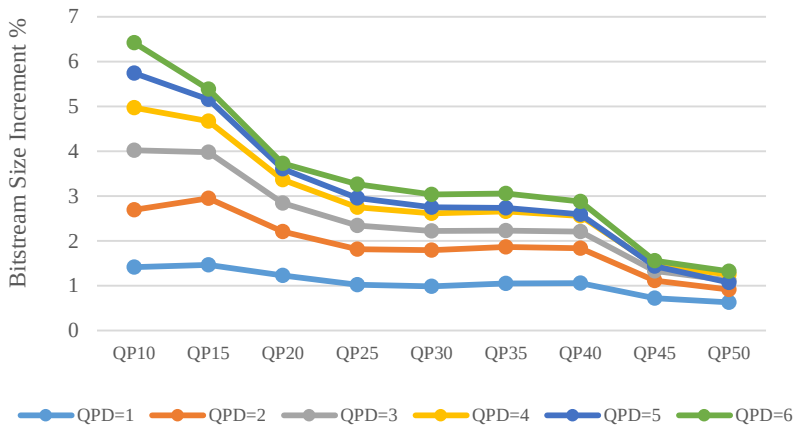


Figura 5.9. Incremento de la tasa de datos generada aplicando diferentes decrementos (QP_D) a distintos valores de QP.

Después de obtener los resultados sobre el incremento asociado a cada pareja de valores de QP/QP_D, se estudió la posibilidad de compensar el incremento en bits provocado por disminuir QP en regiones con texturas homogéneas, aumentando el valor de QP en el resto de áreas de la imagen. Por ejemplo, en el caso de la secuencia *blue_sky* se obtiene la misma calidad subjetiva para QP=31 que para QP=33 con un QP_D=5, lo que además provoca una reducción de la tasa generada de un 18%.

En la Tabla 5.4 se muestra un breve resumen del análisis realizado sobre la secuencia *pedestrian* codificada empleando QP=32, indicando la calidad subjetiva obtenida utilizando la métrica MOS y el incremento en el tamaño del bitstream generado cuando se aplican diferentes valores de QP_D a regiones con texturas homogéneas, respecto a la misma secuencia codificada empleando QP=31 sin aplicar ningún decremento.

Tabla 5.4. Calidad subjetiva y reducción de la tasa de datos generada para la secuencia pedestrian QP=32, empleando diversos decrementos.

QP Value	QP_D	MOS (Ref QP=31)	ΔBitstream (%)
32	2	4.7	-7.83
	3	4.69	-7.07
	4	4.74	-6.28
	5	4.82	-6.12
	6	4.9	-5.57
	7	4.5	-5.40
	8	4.3	-4.97

En la Tabla 5.4 un MOS cercano a 5 indica que la secuencia de vídeo codificada obtiene una calidad subjetiva equivalente a la obtenida para QP=31 sin aplicar decremento. Para el caso de QP=32 y QP_D=6, se obtiene una calidad visual muy similar en ambos casos pero reduciendo el tamaño del fichero generado hasta en un 5.57%. En dicha tabla también se observa cómo decrementos superiores a QP_D=6 suponen un empeoramiento de la calidad subjetiva percibida. Por lo tanto, se puede afirmar que es posible conseguir la misma calidad subjetiva utilizando un decremento de QP en las regiones con texturas homogéneas e incrementando QP en el resto de la imagen, lo que además reduce la tasa de datos generada. Por consiguiente, si se habilita el algoritmo RC y se fuerza los valores correctos de QP_D se podrá obtener una mejora subjetiva de la calidad para una determinada tasa de bits por segundo.

La relación entre QP y el valor de QP a aplicar en regiones con texturas homogéneas (QP_H) es difícil de determinar debido a que depende del número de CU clasificadas como homogéneas. Para simplificar este problema y facilitar la implementación de la solución adoptada, se decidió únicamente aplicar un decremento en las regiones con texturas homogéneas respecto el valor asignado por el RC (QP_R). De esta forma, $QP_H = QP_R - QP_D$. A raíz de esta modificación en el valor de QP asignada, el RC decidirá automáticamente incrementar el valor QP_R en las siguientes CTU y slices para compensar el incremento en el número de bits generados. Se debe de ser extremadamente cuidadoso a la hora de seleccionar los valores de QP_D, debido a que un decremento excesivo provocará que el RC eleve considerablemente los valores de QP_R para alcanzar la tasa de bits objetivo, disminuyendo la calidad visual del vídeo globalmente. Tras realizar numerosas pruebas se observó que las slices tipo intra son más tolerantes a valores de QP_D elevados. Como estas slices son utilizadas como referencias por slices tipo inter, una mejora en la calidad visual de las slices intra provocará una mejora global de calidad en la secuencia de vídeo. Para determinar el valor adecuado de QP_D para cada intervalo de valores de QP_R, es necesario conocer el incremento en la tasa de datos generada mediante un análisis exhaustivo de diferentes tipos de secuencias de vídeo barriendo todos los valores de QP permitidos (0-51) de una manera similar al estudio presentado en la Figura 5.9. Adicionalmente, es necesario comprobar que el RC es capaz de absorber el incremento en bits asociado a QP_D sin elevar en exceso QP_R, para ello se deben analizar los valores de QP asignados por el RC a toda la secuencia y utilizar métricas de calidad para observar decrementos importantes en la secuencia bajo estudio. Este proceso puede automatizarse habilitando el RC y comparando el caso base (QP_D=0) con la propuesta en términos de calidad visual, para diferentes bitrates y tipos de secuencia. Una pérdida drástica de calidad en algún plano de la secuencia reflejará un funcionamiento incorrecto del RC y que el valor de QP_D es

incorrecto. Otro aspecto importante a verificar es que el RC proporcione la tasa de bits objetivo por segundo. En la Figura 5.10 se muestra una comparativa en términos de calidad entre la solución adoptada (en color naranja) aplicando los valores de QP_D finalmente escogidos respecto al caso base (azul) para la secuencia *pedestrian* configurando el RC a un bitrate de 1Mbps.

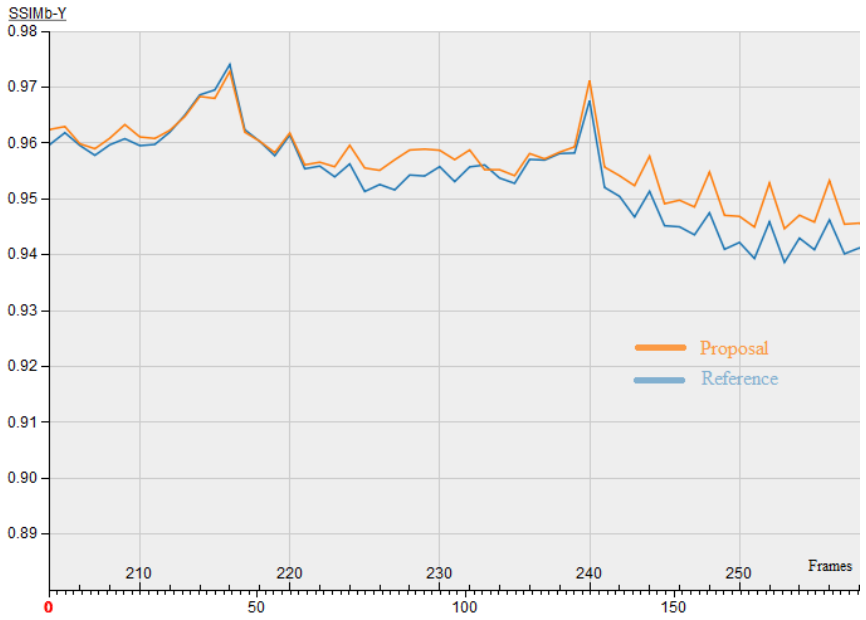


Figura 5.10. Análisis de calidad para la secuencia pedestrian a 1Mbps.

Como se puede apreciar no se producen saltos bruscos de calidad y la solución aporta globalmente una mayor calidad visual que en el caso base sin ninguna modificación. Es importante resaltar que para realizar la medida de calidad visual no se puede utilizar la métrica empleada tradicionalmente PSNR. El uso de PSNR en este caso, carece de sentido, ya que no tiene en cuenta consideraciones perceptuales sobre cómo funciona HVS. Al reducir el valor de QP en regiones con texturas homogéneas estas zonas de la imagen presentarán un valor de PSNR superior al caso base. Por el contrario, como el RC aumenta el valor de QP_R en el resto de la imagen, el valor de PSNR obtenido será inferior. Por lo tanto, el valor de PSNR obtenido para toda una imagen en comparación con el caso base dependerá considerablemente del contenido de la imagen y de la cantidad de texturas homogéneas que presente, pudiendo verse beneficiado o no por el decremento aplicado. Desde el punto de vista de un usuario y teniendo en cuenta el funcionamiento del HVS, la calidad visual subjetiva será mayor ya que los artefactos son percibidos peyorativamente en regiones con texturas homogéneas. Ilustrando lo comentado con un ejemplo, en el caso de la secuencia *blue_sky* se mejorará la calidad percibida en el cielo mientras que se empeorarán las ramas de los árboles, siendo este efecto mejor ponderado globalmente por un usuario utilizando métricas subjetivas como MOS. Si se utilizan métricas con consideraciones perceptuales, incluso siendo métricas objetivas, se puede apreciar la mejora de la calidad visual obtenida tal y como se aprecia para la métrica *Structural Similarity Block Index* (*SSIMb*) en la Figura 5.10. Tras estudiar la literatura existente respecto a las métricas objetivas y realizar diversos experimentos, se decidió utilizar para este análisis SSIMb, ya que está bien correlacionada con la percepción visual humana tal y como se indica en [39]. La Figura 5.11 muestra la dependencia existente entre los valores de QP_D finalmente escogidos y el valor de QP_R utilizado, lo cual conlleva a la existencia de 5 zonas diferentes de QP_D.

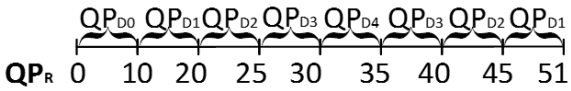


Figura 5.11. Zonas QP_D en función del valor de QP_R utilizado.

En la Figura 5.12 se muestra el mapa de valores de QP obtenidos para la secuencia *in_to_tree* codificada a 512Kbps aplicando decrementos QP_D. Como se puede observar el valor de QP_R para la mayor parte de la imagen es igual a 40, pero en determinadas zonas de la imagen que presentan texturas homogéneas como puede ser el cielo, el valor QP_H se ha reducido a 32 (QP_D = 8).

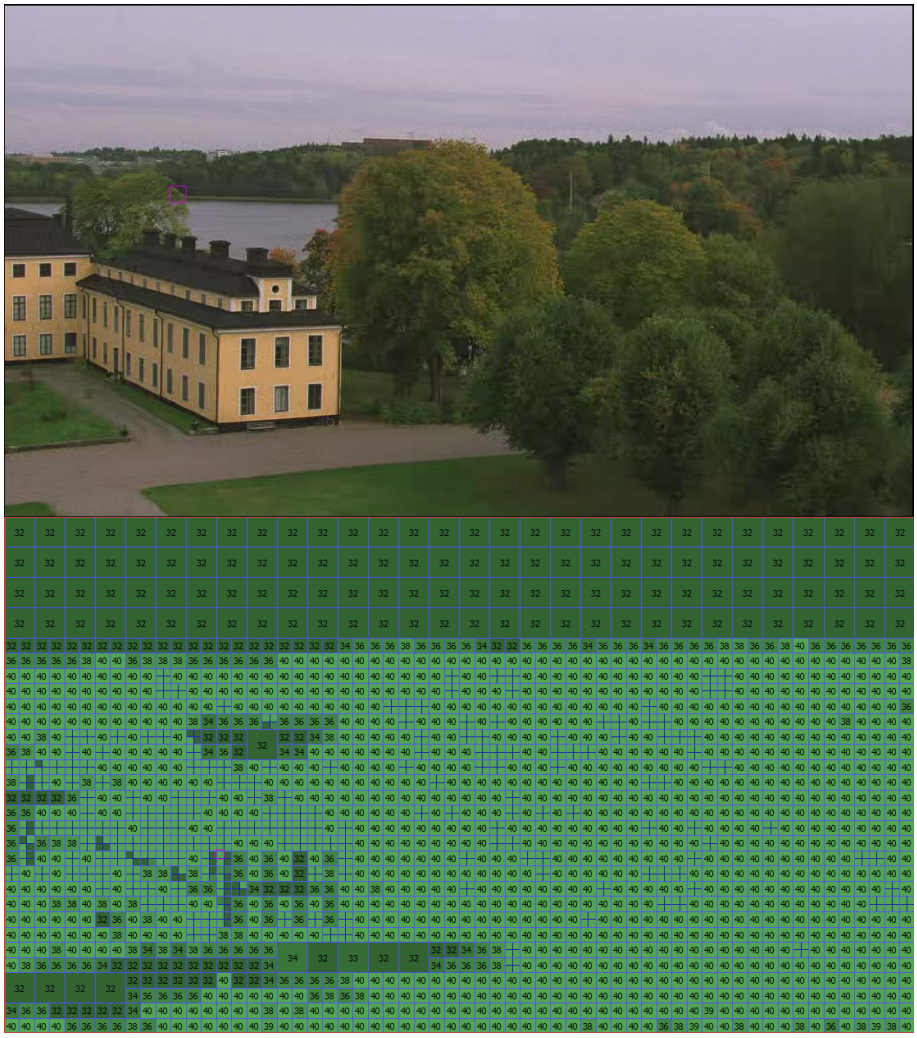


Figura 5.12. Mapa de valores de QP obtenidos en la secuencia in_to_tree @512kbps.

5.3.1.6.2 Eliminación de información redundante

Es ampliamente conocido que la aplicación de algoritmos de pre-procesado puede ser utilizada para mejorar la eficiencia en la compresión de vídeo [40]. Los algoritmos de pre-procesado de vídeo no siguen ningún estándar por lo que pueden ser un factor determinante a la hora de que la calidad proporcionada por un determinado codificador destaque respecto

al resto. En el apartado anterior ya se comentó cómo el HVS es más sensible a percibir detalles en áreas de la imagen que presentan texturas homogéneas como el cielo o un muro de un color plano, por lo que resulta muy desagradable observar artefactos debidos a la codificación en estas zonas. Por otro lado, conforme al funcionamiento del HVS, áreas que presentan muchas texturas pueden ser codificadas con una mayor cantidad de ruido antes que dicho ruido sea apreciable [38]. En [38] los autores mejoran la calidad global de la secuencia de vídeo aumentando el valor de QP en regiones con gran cantidad de texturas, de esta forma se reduce el número de bits destinados a la codificación de estas áreas y el RC destina los bits sobrantes a la codificación de otras zonas de la imagen mejorando su calidad visual por medio de la disminución del valor de QP_R. En el algoritmo presentado en el anterior apartado el enfoque es distinto, se reduce el valor de QP_H en regiones con texturas homogéneas lo que provoca que el RC aumente el valor de QP_R en otras zonas para compensar el aumento del número de bits. Además, el algoritmo se basa en la reutilización de una clasificación basadas en texturas que ha sido realizada para reducir la complejidad del codificador HEVC.

En este apartado, el algoritmo se complementa por medio de un filtrado aplicado sobre las regiones que presentan una distribución caótica, con vistas a reducir la cantidad de detalle que el codificador necesita procesar. Dichas regiones presentan una gran cantidad de coeficientes de alta frecuencia que no son percibidos por el ojo humano y que pueden ser eliminados sin afectar considerablemente a la calidad subjetiva percibida por el usuario. Reduciendo la información no perceptible de la imagen de entrada que el codificador debe procesar se reduce el número de bits que éste emplea y permite destinar estos bits a otras regiones de la imagen. Como se comentó en el apartado anterior, la aplicación de decrementos de QP en regiones con texturas homogéneas provoca un leve aumento en el número de bits generado, que podría ser compensado con la reducción en bits conseguida tras filtrar regiones con texturas caóticas, lo cual aumentará aún más la calidad subjetiva global de la secuencia de vídeo.

La detección de regiones caóticas se realiza para cada píxel de la imagen. El proceso se basa en dos simples condiciones:

1. El píxel no pertenece a una región homogénea. Lo cual se deduce simplemente consultando el mapa de homogeneidad de texturas.
2. Se compara la amplitud del gradiente (A_{GR}) obtenida para el píxel en concreto, con un umbral determinado (GR_{TH}). Una amplitud de gradiente pequeña se asocia a que no existe ninguna direccionalidad.

Si se cumple la primera condición y $A_{GR} < GR_{TH}$, entonces el píxel es clasificado como perteneciente a una región caótica y es sustituido por el valor resultante de aplicar un filtro de mediana de 3x3 píxeles. Debido a que este filtrado puede introducir un efecto de borrosidad (*blurring*) considerable, es muy importante establecer el umbral adecuado. Se decidió hacer uso del gradiente de Sobel debido a que es utilizado en numerosos artículos [2-3, 8, 10-12] para implementar algoritmos de simplificación del proceso de codificación. De esta forma, se podrá utilizar la amplitud del gradiente obtenido para realizar la clasificación de zonas caóticas y para integrar el trabajo realizado con trabajos existentes en la literatura, consiguiendo una reducción extra de la carga computacional.

En la imagen de la derecha de la Figura 5.13 se muestran las diferencias entre la imagen original y la imagen filtrada del primer plano de la secuencia *flower garden*. Las diferencias son resaltadas utilizando el color rojo y azul. Los píxeles de color rojo indican una gran diferencia entre los dos valores de luminancia.

Figura 5.13. Comparación entre la imagen original y la imagen filtrada tras aplicar la detección de regiones caóticas propuesta.

5.3.1.6.3 Implementación de la clasificación de texturas caóticas

La clasificación de texturas caóticas se ha implementado por medio de un kernel de OpenCL que trabaja a nivel de píxel. Básicamente, se accede a un array de amplitudes de gradientes a nivel de píxel obtenido por otro kernel de OpenCL ejecutado previamente para toda la imagen. Se compara el valor de dicha amplitud (A_{GR}) con el umbral preestablecido (GR_{TH}), y se evalúa la condición $A_{GR} < GR_{TH}$. Si se cumple dicha condición, se consulta el mapa binario obtenido por el kernel de clasificación de texturas homogéneas a nivel de bloque 4x4. Si el píxel no pertenece a un bloque con texturas homogéneas, entonces se sustituye por el resultado de aplicar un filtro de mediana 3x3. El filtro de mediana es un filtro no lineal muy simple de implementar que trabaja en el dominio del espacio. Se trata de un filtro estadístico de orden que consiste en sustituir el píxel a tratar por el valor intermedio de la ventana con la que se trabaja (3x3). Se ordenan los valores de los píxeles vecinos seleccionando el valor que queda en el medio. En la Figura 5.14 se muestra un ejemplo de la aplicación de un filtro de mediana 3x3.

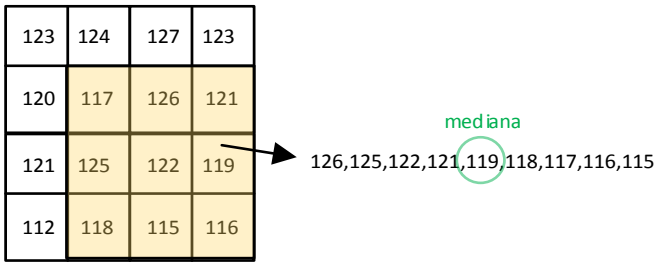


Figura 5.14. Filtro de mediana 3x3.

La única peculiaridad que presenta el kernel en OpenCL implementado para la clasificación caótica es el uso de objetos de tipo *image*. Este tipo de objetos presentan dos importantes ventajas. La primera ventaja es que realiza un control automático de los accesos fuera de la imagen, lo cual resulta muy ventajoso a la hora de simplificar la implementación del kernel, ya que no se requiere realizar un control en los bordes de la imagen. Por otro lado, los objetos de tipo imagen de sólo lectura, utilizan la memoria de texturas disponibles en las GPU, que normalmente trabaja junto a una caché para explotar la localidad espacial que exhiben los patrones de acceso en algoritmos que trabajan con

imágenes. Se trata de una memoria que proporciona un gran ancho de banda efectivo. El resultado de un acceso de lectura fuera de la imagen es configurable por medio del tipo *sampler_t* de OpenCL [41]. En el algoritmo propuesto, si se accede fuera de la imagen se retorna el píxel más cercano perteneciente al borde.

5.3.1.6.4 Reducción de artefactos intrínsecos al HEVC

Como se ha comentado previamente, en el estándar de compresión HEVC se permiten unidades básicas de compresión y transformación mayores a las disponibles en sus predecesores. HEVC permite el uso de CU y TU de tamaño 64x64 y 32x32, mientras que en H264, MPEG4 y MPEG2, el tamaño máximo de macrobloque es 16x16 y el tamaño de transformada a aplicar es normalmente 4x4 (8x8 en H.264 *High Profile*). Esta mayor diversidad permite obtener una importante reducción en el tamaño del bitstream pero también puede provocar errores visuales debidos a su utilización de forma incorrecta. Cuando se aplican los algoritmos de selección de tamaño de bloque contemplando todas las posibilidades que ofrece el estándar y aplicando RDO no aparecen dichos problemas, pero sí están presentes al realizar ciertas simplificaciones en el número de modos de predicción o tamaños de bloque a evaluar. Es habitual en las implementaciones de codificadores HEVC que se limite en función de ciertos algoritmos, el tamaño máximo y mínimo de CU/TU, para reducir la carga computacional asociada a la codificación, siendo estas implementaciones simplificadas susceptibles a la aparición de artefactos. En la Figura 5.15.a, se muestra la aparición de artefactos en ciertas zonas debidos a la codificación. En la Figura 5.15.b, se puede observar que dichos artefactos son debidos al empleo de un tamaño inadecuado de bloque. En los lugares donde aparecen artefactos se está eligiendo un tamaño de bloque que abarca regiones con texturas homogéneas y zonas de la imagen con cierto grado de detalle. El efecto que se aprecia es que en el residuo, se genera ruido en la zona con textura homogénea procedente de la zona que contiene el detalle, tal y como se observa en la Figura 5.15.c. Esto se debe al hecho de aplicar una transformada en regiones de la imagen que contienen coeficientes pertenecientes a baja frecuencia (texturas homogéneas) y medias/altas frecuencias (bordes de la imagen y regiones con detalle). Por lo tanto, una TU de tamaño grande puede llegar a introducir artefactos en los bordes de los objetos contenidos dentro de ella (*ringing artifacts*). Estos artefactos entre sucesivos planos de una misma secuencia, se convierten en el denominado *mosquito noise*, ya que debido al aliasing aleatorio, se percibe un ruido que se observa en el contorno de determinados objetos que se asemeja a un mosquito "revoloteando" alrededor del objeto en cuestión. Estos artefactos son más evidentes a medida que se incrementa el valor de QP, ya que provienen del error introducido en la cuantificación en las componentes de alta frecuencia en el dominio transformado.

Existen filtros para reducir este efecto, conocidos como *deringing filters*, pero son destinados a ser ejecutados sobre la imagen descodificada y no son normativos, por lo que su implementación depende del descodificador utilizado. Esta solución no resuelve el problema en el lado del codificador, así que el bitstream obtenido por el codificador que define el vídeo comprimido presentará los artefactos igualmente, y aparecerán en cualquier descodificador que no implemente este tipo de filtrados.

La principal motivación de la inclusión de SAO en el flujo de codificación de HEVC es paliar este problema. Aunque se ha comprobado empíricamente que SAO reduce estos artefactos considerablemente, no los elimina por completo. Por otro lado, SAO es un proceso computacionalmente muy costoso por lo que en ciertas implementaciones se prescinde de su implementación.

Diversos algoritmos de los que se han desarrollado se basan en realizar paradas prematuras cuando se detecta homogeneidad espacial y/o temporal, lo cual se ha comprobado que conlleva la aparición de este tipo de artefactos.

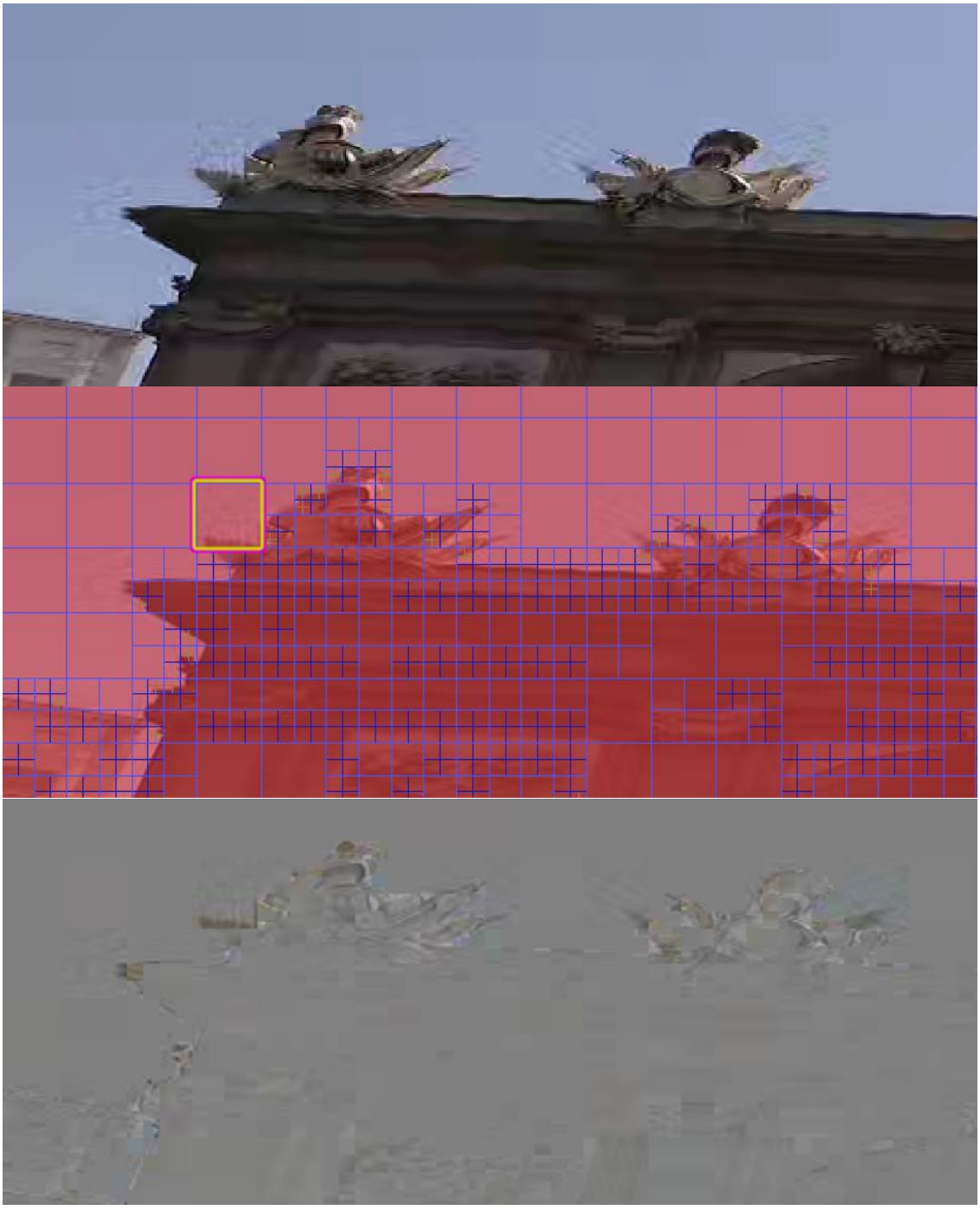


Figura 5.15. Análisis de imagen que presenta artefactos.
a) Imagen con artefactos, b) Tamaños de bloque seleccionados,
c) Residuo obtenido.

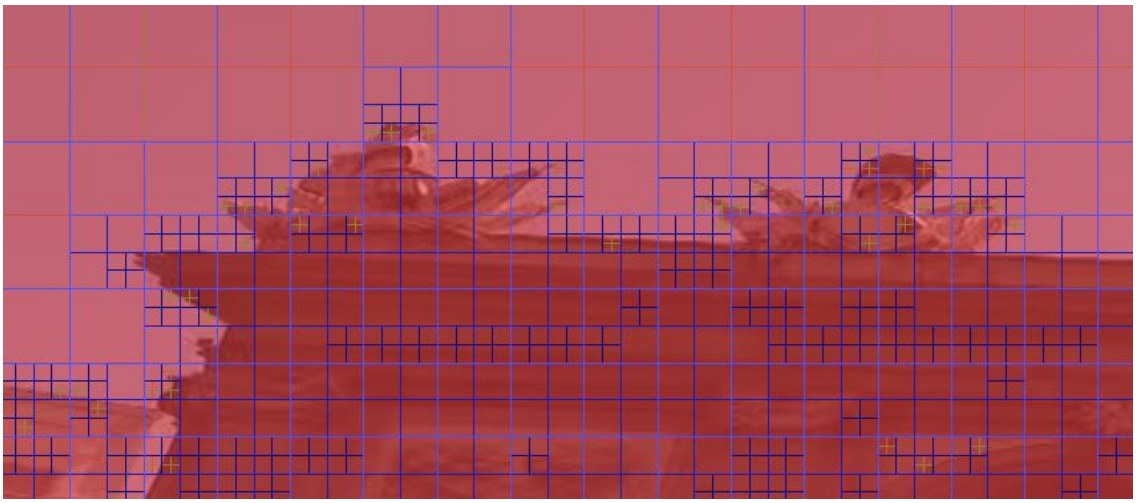
Cuando se habilita el uso de un tamaño mínimo de CU pequeño (8x8 o 4x4), el efecto se reduce, ya que el ruido queda comprendido en la CU de mínimo tamaño que contenga texturas homogéneas y cierto detalle al mismo tiempo. El uso de un algoritmo de división

recursiva de la imagen que siempre contemple la evaluación del tamaño mínimo de CU implica un elevado coste computacional y por supuesto, es contradictorio con las paradas prematuras adoptadas en los algoritmos de simplificación.

Como se ha comentado previamente, el error aparece cuando se selecciona un tamaño de bloque en concreto y en su interior coexisten zonas de imagen con texturas homogéneas y zonas con cierto grado de detalle. Este hecho se agrava haciendo uso de los algoritmos de simplificación que detienen la división recursiva cuando un bloque es considerado como espacial o temporalmente homogéneo, ya que puede ser clasificado de tal forma por ser considerablemente mayor la zona homogénea que la zona que presenta detalle. Para evitar este problema visual sin incrementar enormemente el tiempo de codificación, se ha implementado una solución haciendo uso de la clasificación de texturas homogéneas. Si un bloque contiene dentro de los cuatro sub-bloques que lo componen, sub-bloques con texturas homogéneas y algún bloque que no lo sea. Entonces, se fuerza a que se evalúe el tamaño de bloque inmediatamente inferior ya que se producirán artefactos debidos a la codificación. Esta sub-división se producirá independientemente de que los algoritmos de simplificación diseñados indiquen que no debe de ser dividida la CU por presentar homogeneidad temporal o espacial.

En la Figura 5.16.a se presenta la imagen descodificada obtenida tras aplicar la anterior condición. Como se puede observar mediante una condición relativamente sencilla, se ha conseguido mejorar enormemente el aspecto de la imagen. En la Figura 5.16.b se aprecia cómo en las zonas que anteriormente se veían afectadas, se han utilizado tamaños de bloque inferiores respecto a la selección mostrada en la Figura 5.15.b. En las zonas con texturas homogéneas siguen siendo escogidos con preferencia los tamaños de bloque mayores, por lo que el coste computacional es considerablemente inferior a permitir el uso de tamaños de bloque pequeños en toda la imagen.

a)



b)

Figura 5.16. Resultados tras aplicar la propuesta para la eliminación de artefactos.
a) Imagen descodificada una vez aplicada la condición propuesta,
b) Tamaños de bloque seleccionados una vez aplicada la condición propuesta.

5.3.1.7 Diagrama del flujo de codificación

A continuación, se presenta el diagrama del flujo de codificación propuesto para el codificador HEVC teniendo en cuenta las optimizaciones basadas en un análisis previo de la imagen en busca de regiones que presenten texturas homogéneas (*smooth regions)*. Sobre este tipo de regiones, teniendo en cuenta el tipo de características que presentan, se realizan una serie de simplificaciones con vistas a reducir la carga computacional del codificador sin incrementar excesivamente la tasa de datos generada y sin pérdida apreciable de la calidad visual objetiva. Adicionalmente, se propone una mejora de la calidad visual subjetiva haciendo uso de la clasificación de texturas y teniendo en cuenta el funcionamiento del HVS.

Como se puede observar en la Figura 5.17, para cada CU de las que componen la CTU se consulta el mapa binario asociado a cada tamaño de CU (64x64→depth0, 32x32→depth1, 16x16→depth2 y 8x8→depth3). Si se está procesando una CU de tamaño 64x64 (*Depth==0*) y no presenta texturas homogéneas (*IsSmooth==0*), no se evalúa, pasando directamente a evaluar las cuatro CU de tamaño 32x32 que la constituyen. En el caso de que la CU 64x64 presentase texturas homogéneas, se procedería a ejecutar los algoritmos optimizados de selección de modo de predicción para ese tamaño de bloque y una vez finalizados, se detendría el proceso de división recursivo, procediendo a evaluar la siguiente CTU. Respecto al resto de tamaños de bloque, si una determinada CU es clasificada como espacialmente homogénea, se ejecutan los algoritmos optimizados de selección de modo de predicción para ese tamaño en concreto y se detiene el proceso de división recursivo; procediendo a evaluar la siguiente CU. Si la CU a evaluar es la última CU que compone la CTU, se procedería a evaluar la siguiente CTU.

Los algoritmos de decisión de modo de predicción optimizados permiten reducir el número de modos a evaluar considerablemente. En el caso de la predicción intra, se evaluarán como máximo 5 modos en lugar de los 35 disponibles, siendo éstos el plano, DC, horizontal, vertical y un MPM si éste no está contemplado entre los cuatro anteriores.

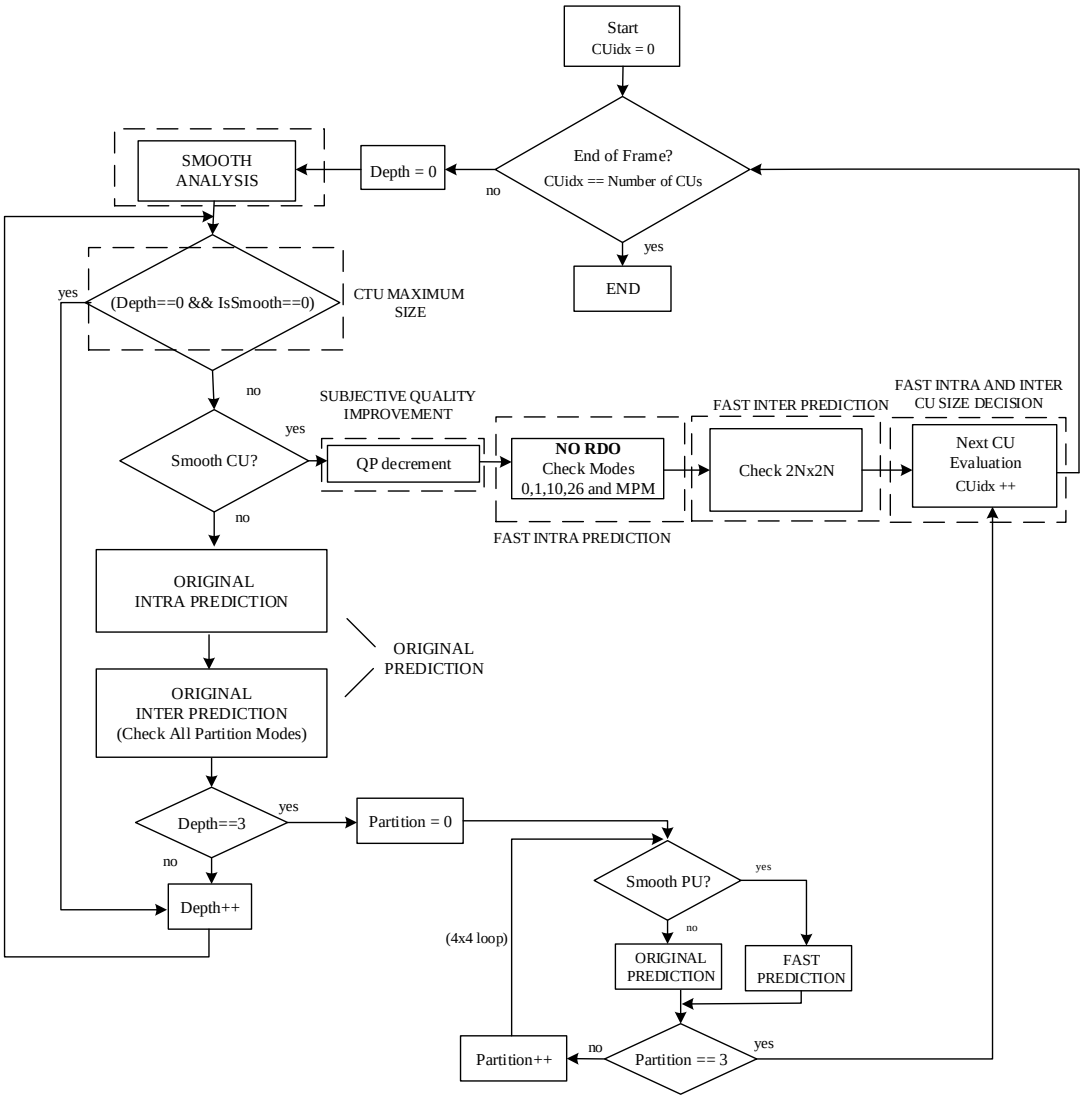


Figura 5.17. Diagrama optimizado del flujo de codificación HEVC basado en un análisis de texturas homogéneas.

Si la slice que se está codificando es de tipo inter, entonces se reducen los modos de partición inter a evaluar de 8 a 1, siendo el único tamaño permitido 2Nx2N, en el que se incluyen los modos de predicción MODE_SKIP y MODE_INTER.
Si el algoritmo RC está activo, para CU con texturas homogéneas se reduce el parámetro QP conforme a un decremento preestablecido, de manera que se mejora la calidad visual en este tipo de regiones y la percepción de calidad subjetiva global, teniendo en cuenta cómo funciona el HVS.

Para las CU que no presentan texturas homogéneas se ejecutan los algoritmos de predicción originales (sin ninguna modificación) y se realiza el proceso de división recursivo. Por ello, el algoritmo propuesto puede ser integrado con otros trabajos existentes en la literatura en la parte del flujo de ejecución asociado a CU espacialmente no homogéneas, permitiendo reducir aún más la carga computacional asociada a la codificación.

5.3.2 Detección de una dirección predominante

5.3.2.1 Análisis de la imagen

Como ya se mencionó en apartados anteriores, existen numerosos trabajos basados en un uso de gradientes para detectar regiones espacialmente homogéneas, puesto que este tipo de algoritmos permiten clasificar las regiones de la imagen en las cuales existe una dirección espacial predominante. En los bloques en los que predomina una dirección espacial, la subdivisión recursiva en tamaños inferiores de bloque no aporta una mejora considerable en el coste RDO. Por otro lado, la mayoría de trabajos relacionados con la optimización del algoritmo de decisión de modo de predicción intra están construidos a partir de cálculos basados en gradientes, tanto en H.264 como para HEVC. Se basan principalmente en el uso de la amplitud del gradiente y del ángulo asociado, lo que proporciona la correlación espacial existente dentro de un bloque de la imagen para estimar de una manera bastante precisa el modo de predicción intra que debería emplearse, ya que normalmente los píxeles en la dirección de un borde local tiene el mismo valor [10]. Por lo tanto, y enfocando el problema de manera inversa, se puede deducir que es posible determinar la dirección predominante en un área de la imagen por medio de los modos de predicción intra que fueron seleccionados.

5.3.2.1.1 Intel's advanced motion estimation extension for OpenCL

Partiendo de la premisa de que es posible determinar si existe una dirección espacial predominante en una imagen analizando los modos de predicción intra seleccionados, se buscaron soluciones existentes que permitiesen, a partir de una imagen de entrada, obtener los modos de predicción intra asociados. La solución óptima se consideró aquella que mediante un módulo de aceleración hardware permita obtener dichos resultados sin coste computacional asociado al procesador encargado de la codificación. La aceleración hardware además conlleva un escaso incremento en el consumo de potencia del sistema de codificación empleado.

Intel proporciona un módulo de aceleración hardware para ser utilizado en H.264, disponible en las GPU embebidas de sus dispositivos [42-43]. La programación de dicho módulo se puede realizar mediante una extensión de OpenCL llamada *Intel's advanced motion estimation extension for OpenCL* [42-44]. El resultado de la ejecución de esta extensión de OpenCL es un conjunto de los modos de predicción intra seleccionados por el algoritmo para cada tamaño de bloque. Los tamaños de bloque que se permiten en H.264 son 16x16, 8x8 y 4x4. Por lo tanto, trabajando a nivel de macrobloque (16x16), el módulo proporcionará un modo de predicción intra 16x16, 4 modos 8x8 y 16 modos de predicción intra 4x4. Este módulo además proporciona el coste asociado a cada tamaño de bloque, por lo que fácilmente se puede deducir qué tamaño de bloque proporciona el menor coste. Es importante resaltar que H.264 únicamente permite seleccionar el tamaño de bloque para cada macrobloque, por lo que todas las particiones son 8x8 o 4x4 si finalmente se escogen alguno de estos tamaños de bloque. Por el contrario, HEVC permite múltiples combinaciones de tamaño de bloque dentro de la misma CTU.

En el algoritmo propuesto, una vez que el módulo hardware ha finalizado su procesado sobre la imagen de entrada, el coste obtenido para cada tamaño de bloque se compara para deducir el tamaño de bloque óptimo, empleando las unidades lógicas de la GPU. Para el tamaño de bloque ganador, si todas las particiones que componen el macrobloque tienen el mismo modo de predicción intra asociado, se considera el macrobloque como espacialmente homogéneo, ya que existe una dirección espacial predominante. Con el

resultado de esta clasificación se construye el correspondiente mapa binario que es compartido con el procesador a cargo de realizar el proceso de codificación.

En el codificador HEVC, dentro del bucle de codificación de CTU se considera una CU como espacialmente homogénea si todos los macrobloques (bloques de 16x16 píxeles) que la componen fueron clasificados de la misma manera. Por lo tanto, en el caso de una CU de tamaño 64x64, es necesario la consulta de 16 bloques 16x16. En el caso de una CU de tamaño 32x32 se requiere la consulta de 4 bloques 16x16, y de únicamente un bloque cuando el tamaño de CU es 16x16. Cuando se evalúa el tamaño de CU igual a 8x8, la CU es clasificada como espacialmente homogénea si el mejor tamaño escogido para H.264 es 8x8 y todos los bloques 4x4 que lo componen tienen el mismo modo de predicción intra. En la Figura 5.18 se ilustra gráficamente este proceso, utilizando un ejemplo en el que dos CU fueron clasificadas como espacialmente homogéneas.

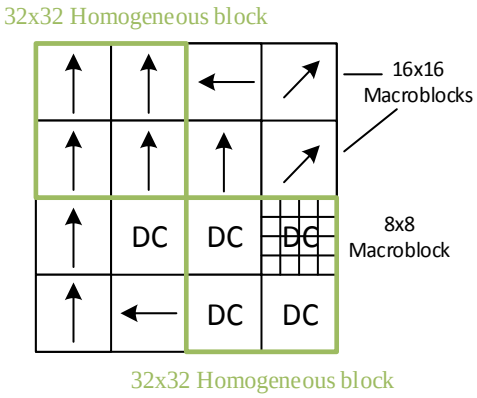


Figura 5.18. Modos de predicción intra H.264 obtenidos para una CTU (64x64) y la clasificación de homogeneidad asociada.

La CU de tamaño 32x32 que aparece en el borde superior izquierdo de la Figura 5.18 fue clasificada como espacialmente homogénea debido a que el mejor tamaño de bloque H.264 escogido por el módulo HW fue 16x16 para los cuatro bloques que conforman la CU 32x32, y el valor de todos los modos de predicción intra asociados fue el modo vertical. En la CU inferior derecha, para los tres macrobloques 16x16 se escogió el modo de predicción intra DC. El macrobloque 8x8, fue clasificado como espacialmente homogéneo debido a que todos los bloques 4x4 que lo componen utilizan el modo de predicción intra DC. Como el modo DC fue el mejor modo de predicción tanto para los tres macrobloques 16x16 como para el macrobloque 8x8, la CU 32x32 que los aglutina se clasificó como espacialmente homogénea.

5.3.2.2 Algoritmo de selección de tamaño de bloque

De manera equivalente al caso de bloques que presentan texturas homogéneas, un bloque que presenta una clara direccionalidad espacial es finalmente clasificado como espacialmente homogéneo. Por lo tanto, el proceso de división recursiva del bloque es detenido, ya que la utilización de tamaños de bloque inferior no supondrá ninguna mejora en las prestaciones de la codificación ya que los bloques de tamaño inferior contenidos presentan características muy similares y se incrementarán los tiempos de ejecución innecesariamente.

5.3.2.3 Selección del modo de predicción intra

En esta sección se propone una nueva optimización, centrada en la reducción de la complejidad asociada a la decisión del modo de predicción intra a aplicar en bloques que presentan una direccionalidad espacial predominante. Como se mencionó previamente, el módulo HW invocado por *Intel's advanced motion estimation extension for OpenCL* proporciona los mejores modos de predicción intra para cada tamaño de bloque permitido en H.264. La predicción intra realizada en el estándar HEVC es muy similar a la realizada en su predecesor, estando basadas ambas en el uso de los píxeles de los vecinos espaciales. Las categorías asociadas a los modos de predicción intra siguen siendo las mismas: Plano, DC, horizontal, vertical y modos angulares (o direccionales). Sin embargo, HEVC especifica la posibilidad de poder utilizar hasta 33 modos direccionales frente a los 8 especificados para H.264. Los modos direccionales permitidos para H.264 son considerados también en HEVC, tal y como se indica en la Figura 5.19.

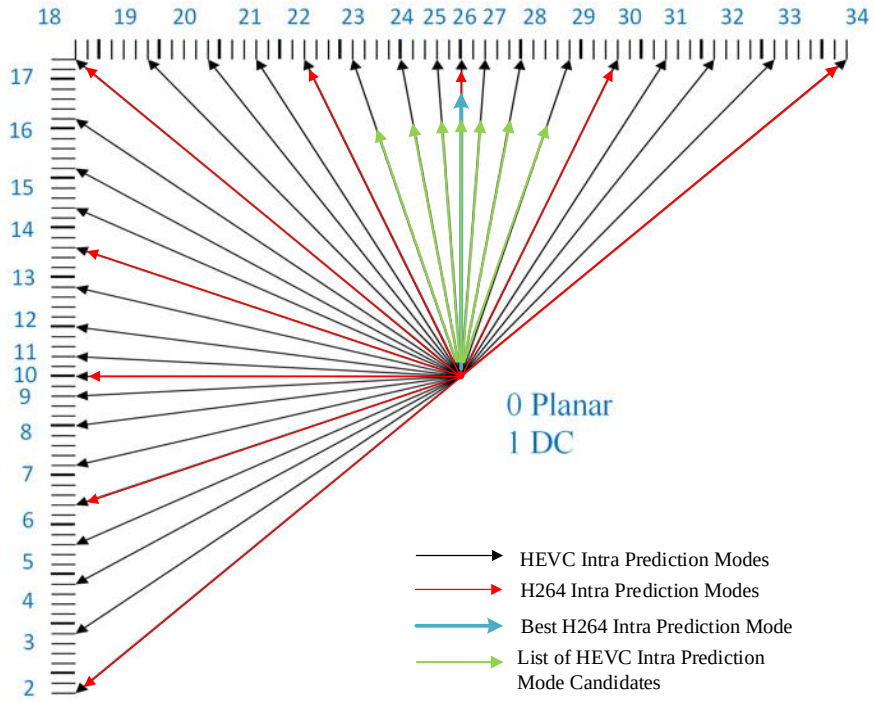


Figura 5.19. Relación entre los modos de predicción intra posibles en H.264 y HEVC.

Para la reducción de la complejidad del algoritmo de decisión de modo de predicción intra en HEVC se propone la reutilización de los modos obtenidos para H.264, reduciendo el conjunto de modos angulares candidatos de 33 a 7 cuando una CU sea clasificada como espacialmente homogénea por tener una direccionalidad espacial predominante. El conjunto de modos candidatos está compuesto por el mejor modo escogido para el estándar H.264 y los 6 modos de predicción angulares más cercanos, los cuales no estaban disponibles para el estándar H.264. Además, los modos no direccionales DC y Plano son siempre incluidos en la lista de candidatos, evaluando hasta 9 modos en lugar de los 35 disponibles en HEVC. Si el mejor modo de predicción en H.264 fue DC o el Plano, únicamente se evaluarán los modos DC y Plano en HEVC.

En la Figura 5.19 se indican los modos de predicción intra que componen la lista de candidatos del algoritmo de decisión de modo del codificador HEVC cuando el modo de predicción intra ganador en H.264 (obtenido por medio del módulo HW) fue el modo vertical (26). En la Figura 5.19 el mejor modo H.264 es resaltado en color azul, los 6 modos HEVC que lo rodean (resaltados en verde) también son añadidos a la lista de candidatos junto al modo DC y Plano, conformando una lista final de 9 candidatos.

5.4 Optimizaciones basadas en la homogeneidad temporal

5.4.1 Análisis de la secuencia de vídeo

Se pueden realizar una serie de optimizaciones sobre el codificador partiendo de la premisa previamente comentada y difundida en la literatura existente de que una CU considerada perteneciente a una región estática, o si en su interior todos los bloques presentan las mismas características relativas al movimiento, ésta puede ser considerada como que presenta homogeneidad temporal y no se requiere su división en bloques de tamaño inferior para caracterizar el movimiento existente.

Para clasificar este tipo de regiones se requiere de un algoritmo de estimación de movimiento que nos permita caracterizar el tipo de movimiento existente en una imagen de una secuencia de vídeo. Este tipo de algoritmos son computacionalmente muy costosos, por lo que el uso de un acelerador HW para realizar el proceso permitirá liberar de una considerable carga computacional al dispositivo a cargo de la codificación propiamente dicha.

En el algoritmo propuesto e implementado, se hace uso de una estimación de movimiento a nivel de bloque 8x8 que trabaja sobre la imagen de entrada. Para realizar la estimación de movimiento, se ha utilizado nuevamente la extensión de OpenCL, Intel Advanced Motion Estimation. Esta extensión proporciona una capa de abstracción de las capacidades del hardware dedicado a la estimación de movimiento disponible en diversas GPU de Intel. De esta forma, tanto las unidades lógicas de la GPU como el procesador encargado de la codificación son liberados de la carga computacional asociada a la estimación de movimiento y pueden ser destinados a otras tareas.

Igual que ocurría para la predicción intra, este módulo está orientado a ser utilizado en el estándar H.264, pero puede emplearse sin problemas para analizar el movimiento de la imagen de entrada, obtener vectores de movimiento a nivel de bloque 8x8 y reutilizar la información para detectar la homogeneidad temporal en la secuencia de vídeo. El módulo hardware es configurado para hacer una búsqueda de movimiento exhaustiva comprobando todos los puntos de una ventana de búsqueda de tamaño [x+/-16, y+-/12] para cada macrobloque que compone la imagen de entrada. La precisión utilizada es de cuarto de píxel y se utiliza la métrica SATD para obtener el coste asociado a los residuos de las diferentes posiciones evaluadas.

La métrica escogida para realizar el cálculo de homogeneidad temporal presente es la desviación media absoluta, ya que esta métrica permite encontrar la media de las desviaciones absolutas de la media de los vectores de movimiento obtenidos en la estimación de movimiento; lo que constituye una métrica robusta para la medida estadística de la dispersión. El valor obtenido para cada componente del movimiento (x e y) es

comparado con un umbral (H_{TH}), siendo clasificado el bloque en cuestión como temporalmente homogéneo si el valor no supera dicho umbral.

En la Ecuación 5.2 se describe cómo calcular la desviación media absoluta para la evaluación de la homogeneidad temporal.

$$Hx = \frac{1}{M}\sum_{k,l}^{M} abs\left(mvx(k,l) - \frac{1}{M}\sum_{k,l}^{M} mvx(k,l)\right). \quad (5.2)$$

Donde M es el número de bloques 8x8 que constituyen la CU que se está evaluando. mvx(k, l) es la componente x del vector de movimiento 8x8 situado en la posición (k, l) dentro de la CU. La misma ecuación es aplicada a la componente Y para obtener la medida de la homogeneidad temporal en el movimiento vertical (H_y). Como se puede observar, para calcular la desviación media absoluta, primero se calculan las medias dentro del mapa de vectores de movimiento a nivel 8x8 y posteriormente, se calculan las desviaciones absolutas de dichas medias. Finalmente, es calculada la media de las desviaciones absolutas.

Aplicando las unidades lógicas de la GPU y un kernel específico creado en OpenCL para permitir interoperabilidad, se crea un mapa de homogeneidad temporal para los tamaños 64x64, 32x32, 16x16 y 8x8 haciendo uso de la Ecuación 5.2 y del umbral H_{TH}.
En la Figura 5.20 se muestra un ejemplo del mapa de vectores de movimiento obtenido para el segundo plano de la secuencia *tractor*. Como se aprecia en dicha figura, varias regiones de la imagen tienen vectores de movimiento similares y son candidatas a ser clasificadas como temporalmente homogéneas. El bloque resaltado en color azul indica un ejemplo de bloque clasificado como temporalmente homogéneo. De manera análoga, se resalta en color amarillo un bloque clasificado como temporalmente no homogéneo.

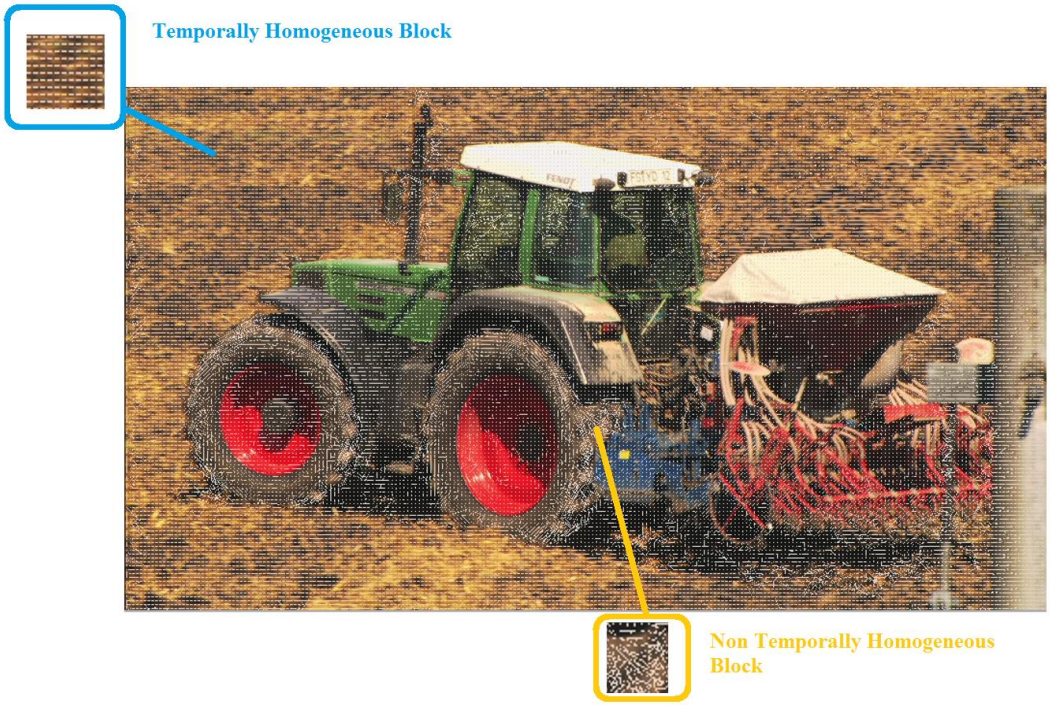


Figura 5.20. Análisis del movimiento para el segundo plano de la secuencia tractor.

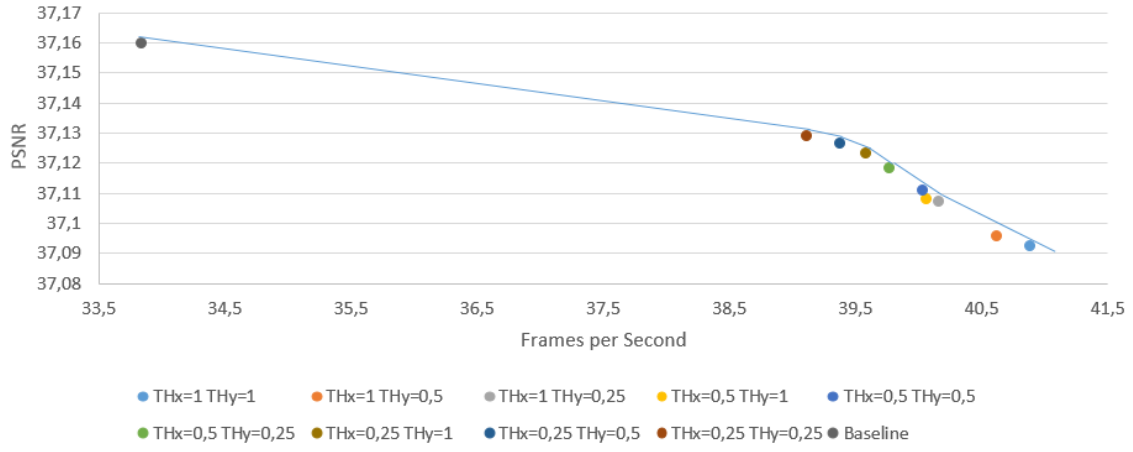
El algoritmo se ha implementado haciendo uso de las capacidades del HW embebido disponible en las GPU de Intel, pero el algoritmo es fácilmente extrapolable a otras

arquitecturas HW. La homogeneidad temporal puede ser calculada haciendo uso de vectores de movimientos obtenidos por cualquier dispositivo, ya sea un co-procesador (FPGA, DSP, GPU, etc.) o el propio dispositivo a cargo de la codificación.

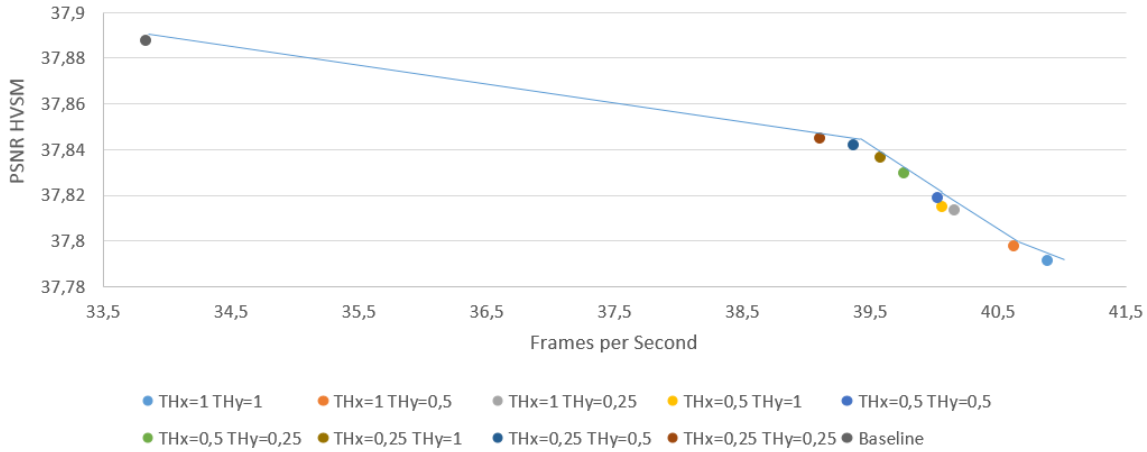
5.4.2 Determinación de los umbrales

Para la determinación de los umbrales se realizó un estudio basado en la PSNR y diferentes métricas que tienen en cuenta el funcionamiento del HVS, en concreto se utilizaron 3-SSIM, MS-SSIM, PSNR, PSNR-HVSM, SSIM, ST-SSIM, VIFP y VQM. El objetivo es obtener una relación de compromiso que permita obtener la mayor calidad visual en el mayor número de métricas y el mayor decremento en tiempos de ejecución. Para resolver el problema relacionado con la búsqueda de los umbrales óptimos se han utilizado diagramas de Pareto comparando calidad y velocidad utilizando diversos valores de TH_x y TH_y a determinados bitrates. Un valor de TH_x igual 0.25 refleja un movimiento equivalente a un cuarto de píxel en horizontal, 0.5 representa movimiento de medio píxel y 1 de un píxel completo. De manera análoga, los valores de TH_y reflejan diferentes magnitudes de movimiento para la componente vertical.

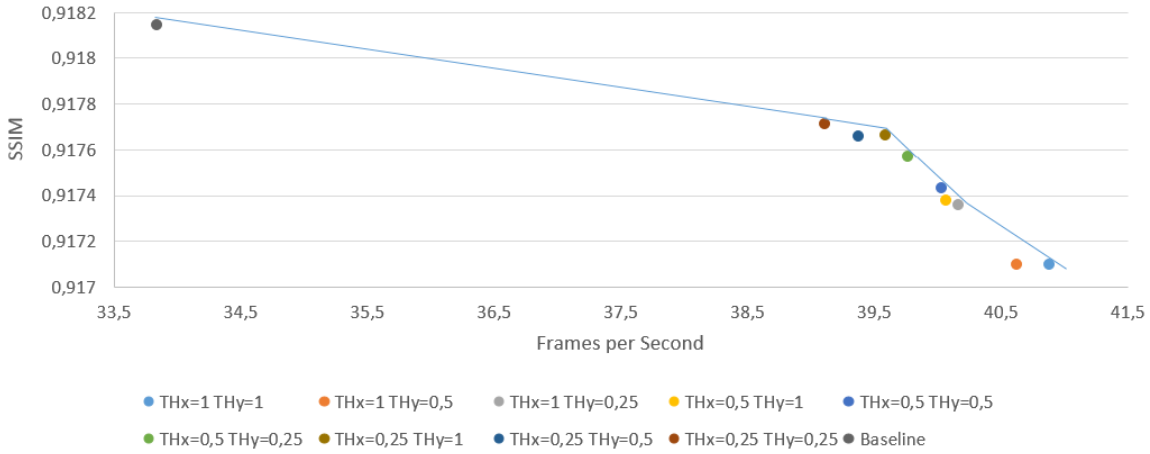
Las diversas gráficas que se muestran en la Figura 5.21, representan los valores obtenidos por las métricas de calidad para distintos valores de TH_x y TH_y configurando el codificador con el RC activo y un bitrate de 10 Mbps. Los resultados que se muestran representan la media de doce secuencias HD: *aspen, blue_sky, crowd_run, ducks_take_off, in_to_tree, park_joy, pedestrian, riverbed, rush_hour, snow_mountain, sun_flower* y *tractor*. En las distintas gráficas cuanto mayor sea el número planos por segundo procesados (*frames per second, fps*), mayor velocidad presenta el algoritmo propuesto y por lo tanto, mayor reducción de tiempos de ejecución se ha conseguido. Para este estudio se trabajó haciendo uso de x265, una implementación de codificador HEVC que permite la ejecución en tiempo real. El caso base se trata del codificador HEVC sin ninguna de las modificaciones propuestas. Como se puede observar en las gráficas de la Figura 5.21, el caso base proporciona la mayor calidad a expensas de menor número de fps.



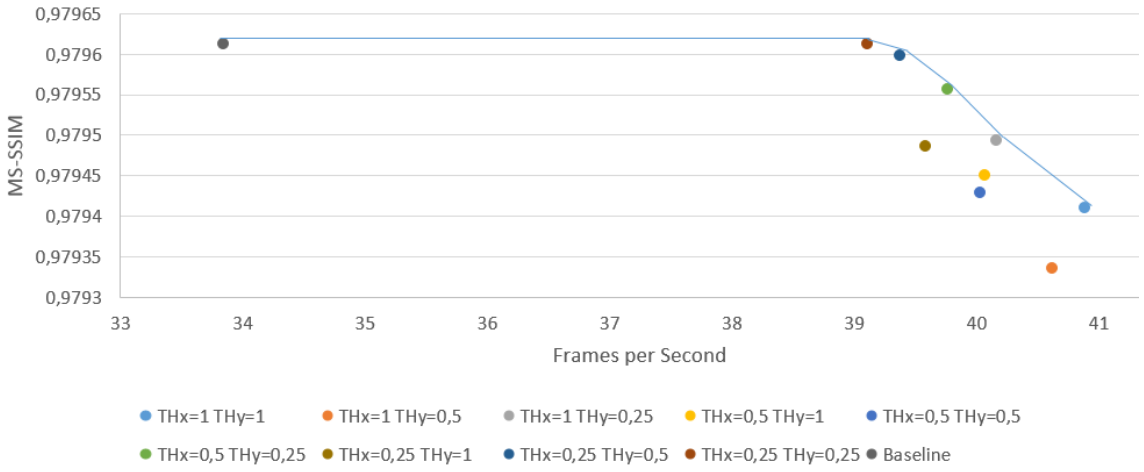
a)


PSNR HVSM
37,9
37,88
37,86
37,84
37,82
37,8
37,78
33,5
34,5
35,5
36,5
37,5
38,5
39,5
40,5
41,5
Frames per Second
THx=1 THy=1
THx=1 THy=0,5
THx=1 THy=0,25
THx=0,5 THy=1
THx=0,5 THy=0,5
THx=0,5 THy=0,25
THx=0,25 THy=1
THx=0,25 THy=0,5
THx=0,25 THy=0,25
Baseline

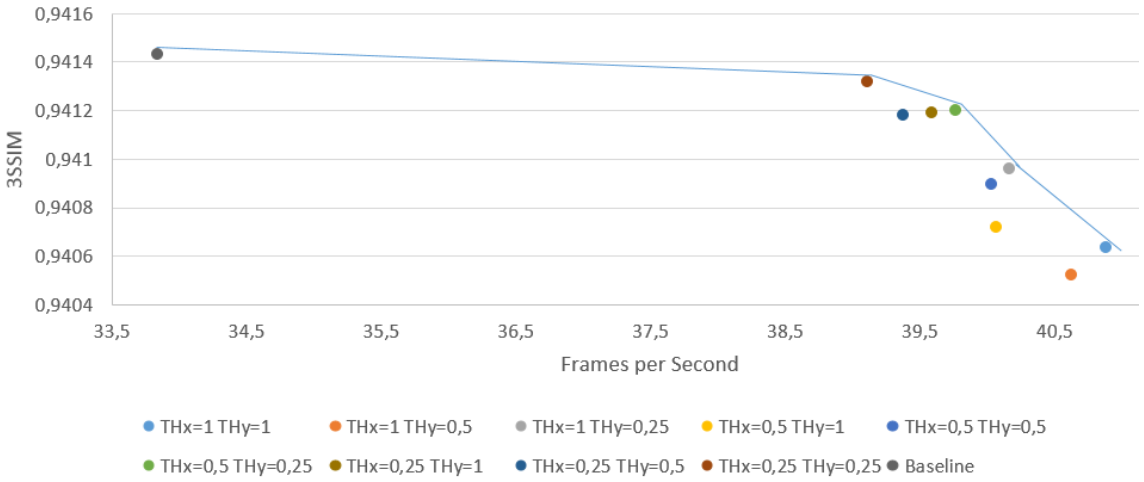
b)


SSIM
0,9182
0,918
0,9178
0,9176
0,9174
0,9172
0,917
33,5
34,5
35,5
36,5
37,5
38,5
39,5
40,5
41,5
Frames per Second
THx=1 THy=1
THx=1 THy=0,5
THx=1 THy=0,25
THx=0,5 THy=1
THx=0,5 THy=0,5
THx=0,5 THy=0,25
THx=0,25 THy=1
THx=0,25 THy=0,5
THx=0,25 THy=0,25
Baseline

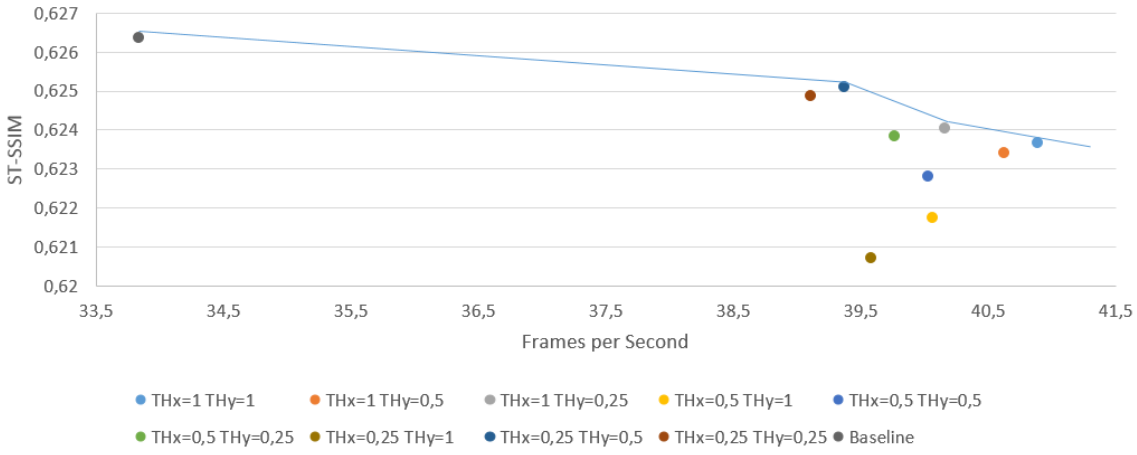
c)


MS-SSIM
0,97965
0,9796
0,97955
0,9795
0,97945
0,9794
0,97935
0,9793
33
34
35
36
37
38
39
40
41
Frames per Second
THx=1 THy=1
THx=1 THy=0,5
THx=1 THy=0,25
THx=0,5 THy=1
THx=0,5 THy=0,5
THx=0,5 THy=0,25
THx=0,25 THy=1
THx=0,25 THy=0,5
THx=0,25 THy=0,25
Baseline

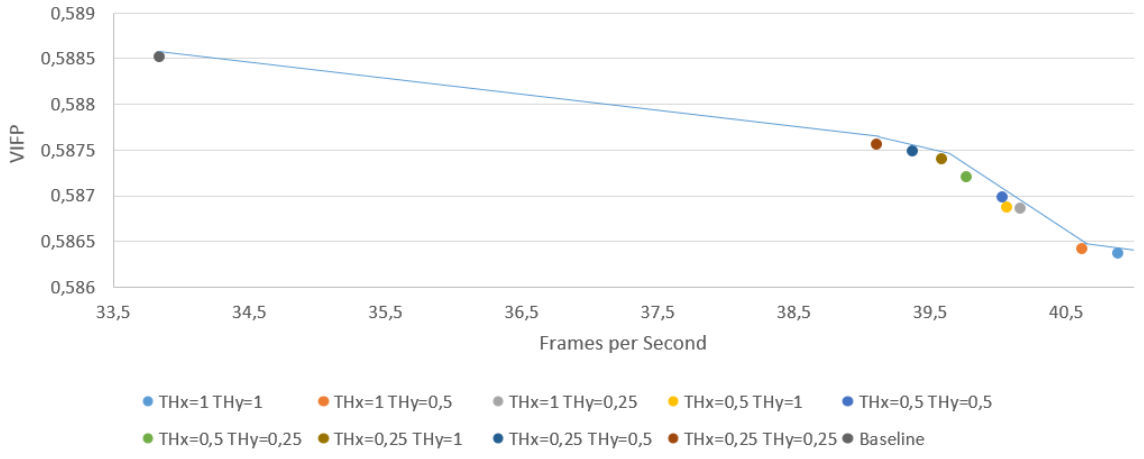
d)


0,9416
0,9414
0,9412
0,941
0,9408
0,9406
0,9404
3SSIM
33,5
34,5
35,5
36,5
37,5
38,5
39,5
40,5
Frames per Second
THx=1 THy=1
THx=1 THy=0,5
THx=1 THy=0,25
THx=0,5 THy=1
THx=0,5 THy=0,5
THx=0,5 THy=0,25
THx=0,25 THy=1
THx=0,25 THy=0,5
THx=0,25 THy=0,25
Baseline

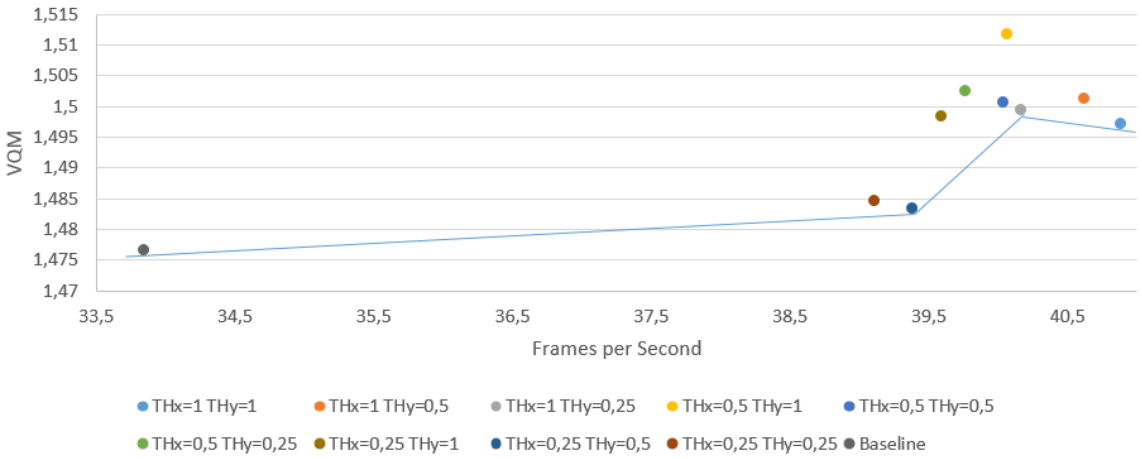
e)


0,627
0,626
0,625
0,624
0,623
0,622
0,621
0,62
ST-SSIM
33,5
34,5
35,5
36,5
37,5
38,5
39,5
40,5
41,5
Frames per Second
THx=1 THy=1
THx=1 THy=0,5
THx=1 THy=0,25
THx=0,5 THy=1
THx=0,5 THy=0,5
THx=0,5 THy=0,25
THx=0,25 THy=1
THx=0,25 THy=0,5
THx=0,25 THy=0,25
Baseline

f)


0,589
0,5885
0,588
0,5875
0,587
0,5865
0,586
VIFP
33,5
34,5
35,5
36,5
37,5
38,5
39,5
40,5
Frames per Second
THx=1 THy=1
THx=1 THy=0,5
THx=1 THy=0,25
THx=0,5 THy=1
THx=0,5 THy=0,5
THx=0,5 THy=0,25
THx=0,25 THy=1
THx=0,25 THy=0,5
THx=0,25 THy=0,25
Baseline

g)



h)

Figura 5.21.Resultados obtenidos para diferentes métricas de calidad a 10 Mbps utilizando diversos umbrales.
a) PSNR, b) PSNR-HVSM, c) SSIM, d) MS-SSIM,
e) 3-SSIM, f) ST-SSIM, g) VFIP, h) VQM.

Las configuraciones óptimas son aquellas que se corresponden con los puntos que pertenecen al frente de Pareto (resaltado con una línea azul en las gráficas de la Figura 5.21. En la Tabla 5.5 se representan aquellas configuraciones que pertenecen al frente de Pareto para diversas métricas de calidad utilizando un bitrate de 10 Mbps.

Tabla 5.5. Configuraciones que pertenecen al frente de Pareto en el espacio Calidad/FPS utilizando varias métricas para un bitrate de 10 Mbps.

Métrica	Valor de los umbrales								
	THx=1 THy=1	THx=1 THy=0.5	THx=1 THy=0.25	THx=0.5 THy=1	THx=0.5 THy=0.5	THx=0.5 THy=0.25	THx=0.25 THy=1	THx=0.25 THy=0.5	THx=0.25 THy=0.25
3SSIM	X		X			X			X
MSSIM	X		X			X		X	X
PSNR	X		X		X	X	X	X	X
PSNR HVSM	X	X	X		X	X	X	X	X
SSIM	X		X		X	X	X		X
StSSIM	X		X					X	
VIFP	X	X	X		X		X	X	X
VQM	X		X					X	

En la Tabla 5.5 se puede observar que las configuraciones que usan los umbrales (TH_x=1, TH_y=1) y (TH_x=1, TH_y=0.25), pertenecen al frente de Pareto para todas las métricas, por lo que pueden considerarse como los valores óptimos de umbrales que se están determinando.

En la gráficas mostradas en la Figura 5.21, se puede observar que la configuración (TH_x=1, TH_y=1) proporciona peor calidad para varias métricas. Por lo tanto, la configuración que hace uso de los valores de umbrales TH_x=1 y TH_y=0.25 puede considerarse como aquella que proporciona mejor relación de compromiso en cuanto a calidad/velocidad de procesamiento se refiere. Por otro lado, en la Figura 5.21 se puede observar cómo algunas métricas como PSNR no son de utilidad para el análisis, ya que casi todos los puntos pertenecen al frente de Pareto, mientras que otras métricas como VQM o ST-SSIM permiten definir fácilmente qué puntos son los dominantes.

Siguiendo el proceso de análisis previamente comentado, con el objetivo de determinar los valores definitivos de los umbrales, se realizó un estudio más exhaustivo empleando mayor número de secuencias y configurando el RC para utilizar bitrates comprendidos entre 1 y 20 Mbps.

5.4.3 Optimizaciones

5.4.3.1 Algoritmo de selección de tamaño de bloque

En las CU de tipo inter, si la CU fue clasificada como temporalmente homogénea, se detiene el proceso de subdivisión recursiva, puesto que la utilización de tamaños de bloques más pequeños no aportará mayor precisión a la hora de definir el movimiento en este tipo de áreas.

5.4.3.2 Selección del modo de partición inter

De manera equivalente al estudio realizado para regiones con texturas homogéneas, se realizó un análisis de los modos de partición inter más probables en regiones temporalmente homogéneas, alcanzando las mismas conclusiones. Por ello, el número de modos de partición inter a evaluar en zonas temporalmente homogéneas es reducido a únicamente el modo 2Nx2N.

5.4.3.3 Configuración adaptativa de la ventana de búsqueda

Tal y como se ha comentado previamente, si una CU es clasificada como temporalmente homogénea, la CU en cuestión pertenece a una región estática o en su interior todos los bloques que la componen presentan un movimiento similar. La Ecuación 5.2 proporciona una aproximación del tipo de movimiento que una CU posee, por lo que haciendo uso de la media de los vectores de movimiento que se ha utilizado para obtener la desviación media absoluta, es posible derivar si se trata de una secuencia con gran cantidad de movimiento o no. Regiones de la imagen que sean muy dinámicas requerirán de una amplia ventana de búsqueda para que la estimación de movimiento sea capaz de determinar de una manera precisa los vectores de movimiento. Por el contrario, en regiones con escaso movimiento (semi-estáticas o estáticas) se podrá reducir la ventana de búsqueda para caracterizar dicho movimiento. Cuanto menor sea la ventana de búsqueda, menos posiciones evaluará la estimación de movimiento con el correspondiente ahorro computacional que esto supone.

La media es calculada utilizando el mapa de vectores de movimiento obtenido por una estimación de movimiento basada en el estándar H.264 aplicada sobre la imagen de entrada en la etapa de pre-análisis, en lugar de aplicarla sobre la imagen reconstruida, pero puede ser utilizada como una aproximación del movimiento existente en la secuencia de vídeo. Esta aproximación permite derivar el movimiento presente dentro de la imagen reconstruida que se utiliza para la estimación de movimiento en el bucle de codificación de CTU en el estándar HEVC.

En la propuesta realizada en la presente Tesis, se utiliza la media de los vectores de movimiento obtenidos por la estimación de movimiento hardware basada en H.264 (MV_{median}) para establecer el tamaño de la ventana de búsqueda utilizada en la estimación

de movimiento del codificador HEVC en CU clasificadas como temporalmente homogéneas, tal y como se indica en la Figura 5.22.

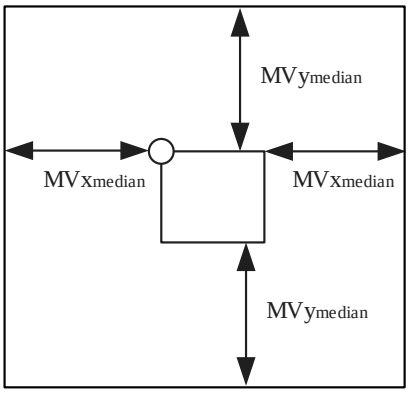


Figura 5.22. Ventana de búsqueda adaptativa de la estimación de movimiento para CU temporalmente homogéneas.

Por lo tanto, el rango de la zona de búsqueda será MVx_{median} x MVy_{median} estando centrada en las coordenadas definidas por el vector de movimiento predicho (MVPx, MVPy). El vector de movimiento predicho es calculado según el procedimiento definido en el estándar HEVC [45] utilizando los parámetros que definen el movimiento de PU vecinas. La estimación de movimiento utilizada es la definida en el software HM v16.2 pero implementando un proceso RDO optimizado en el que se utiliza la imagen de entrada en lugar de la imagen reconstruida para calcular la parte asociada a la distorsión en la función de coste RDO. Esto reduce considerablemente el gasto computacional del cálculo del coste RDO, puesto que no es necesario realizar todos los procesos implicados en la obtención de la imagen reconstruida para cada posición evaluada en la estimación de movimiento.

5.4.3.4 Parada prematura aplicada en la predicción inter

El módulo de hardware dedicado que se ha utilizado para implementar la estimación de movimiento y la decisión de modo de predicción intra basados en el estándar H.264, proporciona los valores asociados a las funciones de coste de los mejores modos de predicción intra/inter H.264. Aunque HEVC incluye mejores métodos que H.264 para la obtención de los modos óptimos de predicción intra/inter, están ampliamente basados en los utilizados en H.264. Por lo tanto, los resultados obtenidos en H.264 pueden ser reutilizados para proporcionar ciertos indicadores de cuál será el mejor modo de predicción en HEVC. Si el coste del mejor modo de predicción intra asociado a un macrobloque H.264 es muy alto en comparación con el mejor modo de la predicción inter, puede derivarse que es muy probable que en HEVC el modo ganador utilice predicción intra.

Por lo tanto, en la propuesta realizada, el coste proporcionado por el módulo hardware para el mejor modo de la predicción inter H.264 es comparado con el obtenido para el mejor modo de predicción intra H.264 multiplicado por cierto umbral (TH_{SAD}). Si se cumple la condición $SAD_{H264inter} > SAD_{H264intra} * TH_{SAD}$, entonces se considera que la predicción inter no debe de ser evaluada en el codificador HEVC, eliminando el enorme coste computacional asociado a la estimación de movimiento y otros procesos involucrados en la predicción inter. Como HEVC permite CU de tamaño 64x64 y 32x32, los cuales no

están permitidos en H.264, los costes para los tamaños superiores a 16x16 son obtenidos sumando los costes de los bloques 16x16 que componen la CU correspondiente.

El umbral TH_{SAD} fue determinado empíricamente siguiendo un proceso similar al utilizado para la determinación de TH_x y TH_y explicado en la sección 5.4.2. El valor de TH_{SAD} ha sido validado en la sección 5.6, correspondiente a los experimentos.

5.5 Diagrama de flujo global

En este apartado se muestra el diagrama de flujo de las contribuciones descritas a lo largo de este capítulo que han sido implementadas en un codificador conforme al estándar HEVC. La mayor parte de las optimizaciones realizadas están basadas en un análisis de la imagen para detectar regiones que presentan homogeneidad espacial y/o temporal. Sobre este tipo de regiones se aplican una serie de algoritmos que permiten reducir la complejidad del proceso de codificación con un leve incremento en la tasa de datos generada y sin pérdida apreciable de calidad visual. Adicionalmente, se han implementado una serie de algoritmos que permiten mejorar la calidad subjetiva percibida por el usuario final que visualice el vídeo descodificado.

En la Figura 5.23 se puede observar una representación gráfica de los procesos que son ejecutados en el co-procesador (en las implementaciones realizadas, los modelos de GPU embebida de Intel, HD Graphics 530 y HD Graphics 4600).

Cada imagen perteneciente a la secuencia de vídeo es sometida a un análisis ejecutado en el pre-procesador. Los primeros procesos que son realizados son la estimación de movimiento y el algoritmo de selección de modo de predicción intra bajo el estándar H.264, procesos que en la propuesta realizada son ejecutados haciendo uso de módulos hardware destinados específicamente a este fin en las GPU de Intel. De esta forma, las unidades lógicas de la GPU quedan libres para ejecutar los posteriores análisis de la imagen que se indican en la Figura 5.23. El coste computacional de estos procesos es practicamente cero para el procesador encargado de realizar la codificación propiamente dicha. El resultado de la estimación de movimiento será un mapa de vectores de movimiento junto con el coste asociado a dicho vectores a nivel de bloques 8x8. El algoritmo de selección de modo de predicción intra proporcionará los mejores modos de predicción intra para cada tamaño de bloque disponible en H.264 (16x16, 8x8 y 4x4) y los costes asociados. Toda esta información será utilizada posteriormente por las unidades lógicas de la GPU.

El siguiente proceso que se ejecuta es el análisis de la imagen en busca de regiones con texturas homogéneas haciendo uso del ratio DC, obteniendo un mapa de regiones homogéneas para cada tamaño de bloque permitido en HEVC (64x64, 32x32, 16x16, 8x8 y 4x4). Este proceso es ejecutado haciendo uso de las unidades lógicas de la GPU.

A continuación, se ejecuta la clasificación de regiones espacialmente homogéneas que presentan una clara direccionalidad espacial. Para ello, se hace uso de los datos obtenidos tras ejecutar el algoritmo de selección de modo de predicción intra H.264.

El flujo de ejecución continúa aplicando el análisis de la homogeneidad temporal sobre la señal de vídeo de entrada utilizando las unidades lógicas de la GPU. Dicho análisis, se basa en el cálculo de la desviación media absoluta utilizando el mapa de vectores de movimiento obtenido por la estimación de movimiento HW H.264. Como resultado de este análisis, se obtiene un mapa que define la homogeneidad temporal presente para cada tamaño de bloque permitido en HEVC.

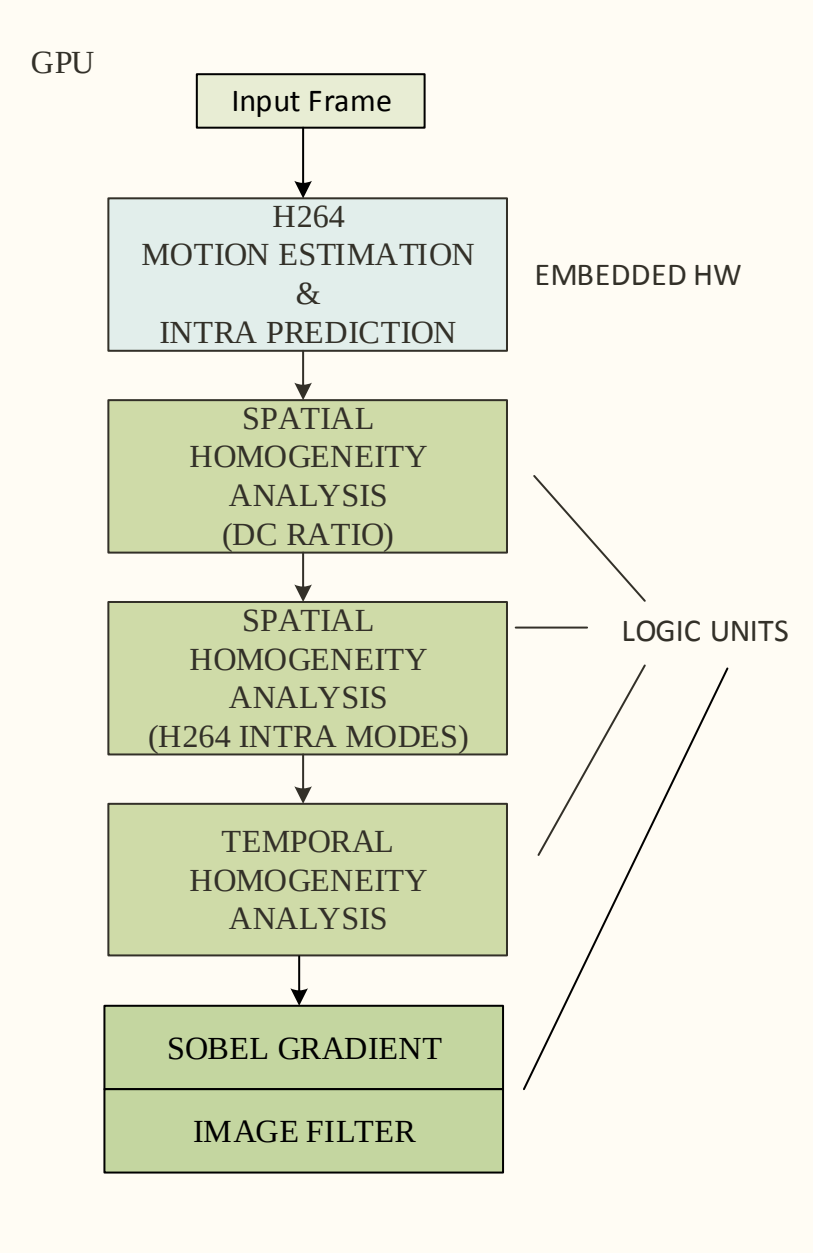


Figura 5.23. Procesos de pre-análisis de la imagen ejecutados en el pre-procesador.

El filtrado sobre la imagen de entrada es aplicado para bajos bitrates en regiones que presentan estructuras caóticas, haciendo uso de la clasificación que indica las regiones con texturas homogéneas y de la amplitud del gradiente. El cálculo de la amplitud del gradiente se realiza en un kernel independiente. En este kernel se puede habilitar la obtención del ángulo del gradiente si va a usar utilizado en posteriores optimizaciones. Como se verá en el apartado de resultados, se realizaron experimentos integrando algunas de las optimizaciones realizadas con métodos propuestos en otros trabajos para comprobar la compatibilidad e interoperabilidad de las mejoras propuestas. El trabajo utilizado en cuestión, hace uso tanto de la amplitud como del ángulo del gradiente, que en nuestro caso, es calculado haciendo uso de las unidades lógicas de la GPU. El filtrado aplicado sobre la imagen de entrada es imperceptible para el HVS y permite reducir la tasa de datos generada.

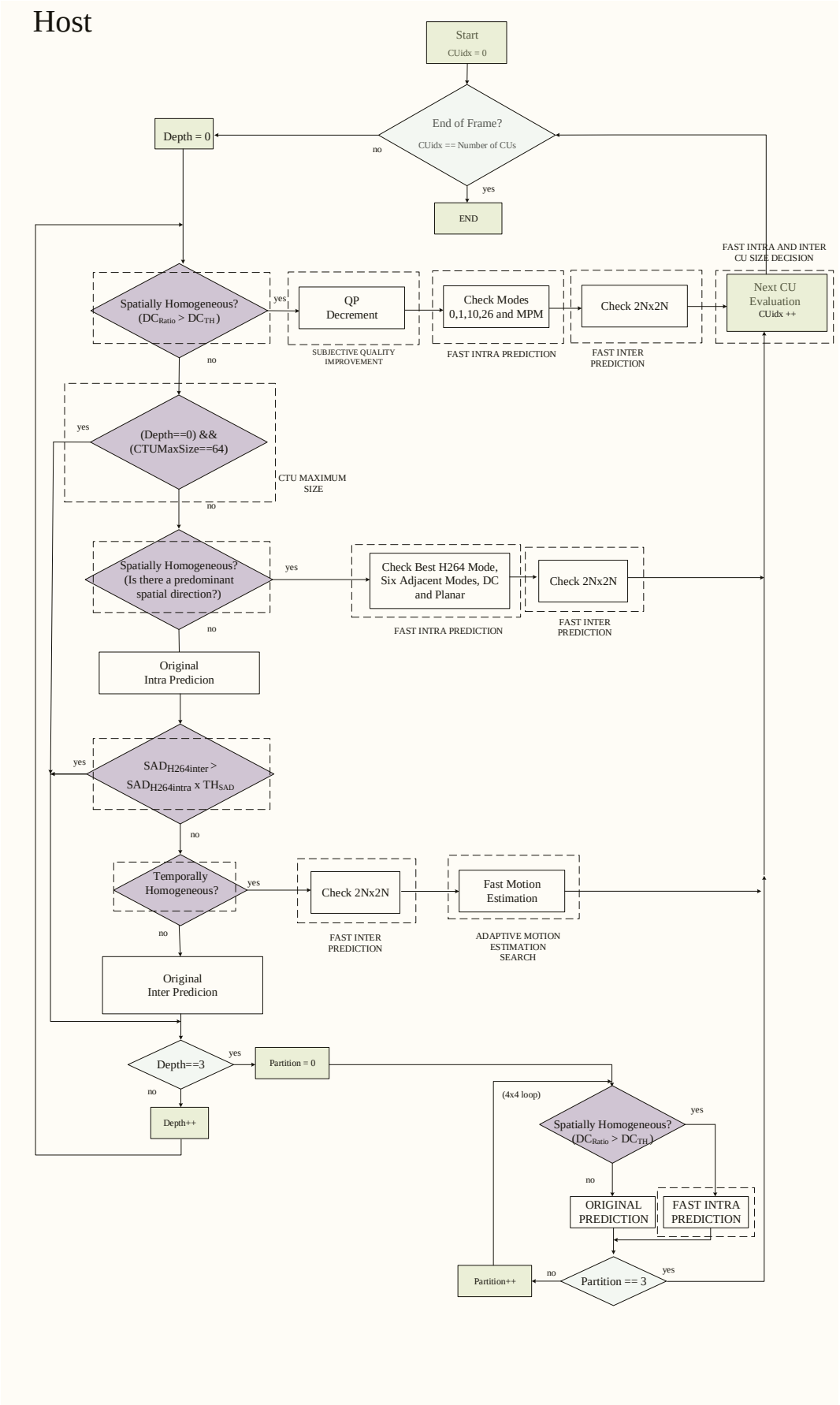


Figura 5.24. Diagrama de flujo de codificación HEVC optimizado.

En la Figura 5.24 se muestra el diagrama de flujo propuesto para reducir la carga computacional asociada a la codificación HEVC. Para cada CU, se consulta el mapa asociado a la clasificación de texturas homogéneas. Si la CU fue clasificada como espacialmente homogénea porque presenta texturas homogéneas y en el caso de que el RC esté activo, el valor de QP es reducido para mejorar la calidad visual subjetiva percibida. Posteriormente, se aplica un algoritmo de decisión de modo de predicción intra simplificado en el que únicamente se evalúan los modos de predicción Plano, DC, Horizonal, Vertical y MPM (si el modo más probable no es uno de los cuatro anteriores); en lugar de los 35 que permite el estándar HEVC. Adicionalmente, se elimina la utilización de la función de coste RDO. Si la CU que se evalúa pertenece a una slice tipo inter, se ejecuta un algoritmo de predicción inter optimizado en el que únicamente se evalúa el modo de partición inter 2Nx2N (que incluye el modo de predicción SKIP), por lo que el número de modos de partición a evaluar se reduce de 8 a únicamente 1.

Si la CU no presenta texturas homogéneas y el tamaño de bloque actual es 64x64, se procede a evaluar el siguiente tamaño de bloque ya que sólo se permite este tamaño en regiones con texturas homogéneas. A continuación, se evalúa si la CU fue clasificada como espacialmente homogénea por existir una direccionalidad espacial predominante. En caso afirmativo, se aplica un algoritmo de decisión de modo de predicción intra optimizado. Básicamente, se evalúa el modo ganador de la predicción intra H.264 y los seis modos adyacentes HEVC que no existen en H.264. Además, se incluyen en la lista de candidatos los modos DC y Plano por su alta probabilidad de ocurrencia. El algoritmo de predicción inter optimizado que se ejecuta para este tipo de CU es el mismo que el utilizado para CU que presentaban texturas homogéneas.

Si la CU no fue clasificada como espacialmente homogénea porque no presenta ni texturas homogéneas ni una direccionalidad espacial predominante, entonces se ejecuta la predicción intra original sin ninguna modificación.

Posteriormente, se evalúa el coste asociado a la predicción intra y a la predicción inter H.264 que se obtuvo por medio del uso de módulos HW dedicados. Si se cumple la condición $SAD_{H264inter} > SAD_{H264intra}$ x * TH_{SAD}, entonces no se evaluará la predicción inter HEVC debido a que elegir este tipo de predicción para la CU no aportará mejora respecto al uso de la predicción intra.

Si no se cumple la condición, entonces se requiere la evaluación del coste asociado a la predicción inter HEVC. En el caso de que la CU haya sido categorizada como temporalmente homogénea por el algoritmo de clasificación, entonces se ejecuta un algoritmo de predicción inter optimizado evaluando únicamente el modo de partición inter 2Nx2N. Adicionalmente, para este tipo de partición inter se ejecuta un algoritmo optimizado de estimación de movimiento en el que la ventana de la zona de búsqueda se adapta al movimiento existente en la secuencia de vídeo. Para ello, se hace uso de los vectores de movimiento obtenidos por la estimación de movimiento HW de H.264.

Cuando una CU es clasificada como espacialmente y/o temporalmente homogénea, se detiene la división recursiva de la CU en bloques de tamaño inferior conforme al algoritmo de decisión de tamaño de bloque optimizado que se ha presentado en este trabajo. El bucle que se muestra en la parte inferior de la Figura 5.24 se corresponde con el bucle que se ejecuta para las cuatro PU 4x4 existentes dentro de una CU 8x8 tipo intra. En este bucle sólo se evalúa si la PU presenta texturas homogéneas, si es así se ejecuta el algoritmo de decisión de modo intra optimizado. En caso contrario, se ejecuta la predicción intra original.

Si la CU no es clasificada como espacial o temporalmente homogénea, entonces se prosigue con el bucle recursivo de división en bloques de tamaño inferior. A continuación,

se evaluará para cada tamaño de CU permitido las condiciones antes expuestas, y ejecutando en los casos que proceda los algoritmos de optimización presentados.

5.6 Resultados

Los resultados obtenidos tras la integración de los diferentes algoritmos propuestos en el flujo de codificación HEVC han sido publicados en diversos artículos [45-52]. En los siguientes apartados se procederá a describir los resultados obtenidos para los diferentes algoritmos que han sido descritos con anterioridad relacionándolos con las publicaciones correspondientes.

5.6.1 Optimizaciones basadas en la homogeneidad espacial existente en texturas homogéneas

5.6.1.1 Algoritmo de selección de tamaño de bloque

En el trabajo que se detalla en [52] se presenta por primera vez el algoritmo de selección de tamaño de bloque basado en la detección de regiones con texturas homogéneas. En este trabajo la optimización se centra en la reducción de la complejidad de la codificación HEVC en secuencias de resolución UHD implementando la clasificación de texturas homogéneas basada en el ratio DC en el mismo procesador que se encarga de la codificación. En el caso de que una CU contenga texturas homogéneas se detiene el proceso recursivo de división en bloques de tamaños inferiores. En las Tablas 5.6 y 5.7 se muestran los resultados obtenidos en términos de incremento del tamaño de bitstream generado, incremento del valor de PSNR, incremento en el tiempo de codificación e incrementos en el tiempo dedicado a la predicción intra/inter. La verificación se realiza con el algoritmo RC deshabilitado para independizar la codificación de la selección de QP que realice el RC, analizando para un determinado nivel de calidad prefijado por el valor de QP cómo varían las prestaciones del codificador. Los resultados que se muestran se corresponden a la media de los resultados obtenidos utilizando nueve valores de QP diferentes: 10, 15, 20, 25, 30, 35, 40, 45 y 50. Para la obtención de los resultados se utilizó el software de referencia HM v16.2 empleando las configuraciones intra-only y low-delay P, cuyos ficheros de configuración se proporcionan junto a la distribución del software HM.

Como se puede observar en las Tablas 5.6 y 5.7, se han alcanzado reducciones considerables en el tiempo de ejecución, logrando una reducción de hasta 46.25% en el mejor caso. Por lo general, el tamaño del bitstream es levemente reducido, siendo 0.53% la mayor reducción y 0.56% el mayor incremento. La reducción en términos de calidad es prácticamente imperceptible, siendo 0.2 dB la mayor pérdida de calidad.

Tabla 5.6. Resultados del algoritmo de selección de tamaño de bloque para la configuración intra-only.

Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔEncoding Time (%)	ΔIntra Prediction Time (%)
blue_sky 1080p25	-0.14	-0.04	-40.61	-42.19
pedestrian_area 1080p25	-0.02	-0.05	-41.47	-42.98
riverbed 1080p25	-0.11	-0.01	-12.68	-13.43
rush_hour 1080p25	-0.53	-0.05	-46.25	-47.93
shields 720p59	-0.07	-0.02	-6.67	-6.99
stockholm 720p60	-0.09	-0.03	-17.33	-17.80

Tabla 5.7. Resultados del algoritmo de selección de tamaño de bloque para la configuración low-delay P.

Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔEncoding Time (%)	ΔIntra Prediction Time (%)	ΔInter Prediction Time (%)
blue_sky 1080p25	-0.11	-0.15	-29.52	-18.41	-31.66
pedestrian_area 1080p25	0.56	-0.15	-24.67	-30.11	-25.23
riverbed 1080p25	-0.06	-0.09	-6.12	-16.99	-1.24
rush_hour 1080p25	-0.14	-0.20	-27.19	-38.13	-28.73
shields 720p59	0.11	-0.11	-12.31	-28.2	-13.81
stockholm 720p60	-0.27	-0.07	-4.15	-18.37	-2.07

5.6.1.1.1 Integración con trabajos existentes

Una de las bondades que presentan los algoritmos desarrollados, es la posibilidad de coexistencia con otros trabajos existentes en la literatura. Para demostrar esta afirmación se integró el algoritmo de selección de tamaño de bloque basado en la detección de texturas homogéneas con un método propuesto en un trabajo ya existente. Dado que existen multitud de algoritmos basados en el cálculo de gradientes para reducir la complejidad de la codificación HEVC, se optó por integrar el trabajo realizado con un algoritmo de este tipo. Tras realizar una lectura exhaustiva de la literatura existente, se decidió utilizar para este propósito el algoritmo denominado *Edge-based Intra Coding Unit Size Decision* expuesto en [10], ya que ofrece excelentes prestaciones. El algoritmo es únicamente aplicado en regiones que no presentan texturas homogéneas, de manera que ambos algoritmos se complementan, ya que se consigue una reducción de tiempos de ejecución en las regiones no clasificadas por nuestro algoritmo como espacialmente homogéneas y por otro lado, se reducen el número de operaciones requeridas en [10] al aplicar el algoritmo basado en gradientes sólo en determinadas regiones de la imagen en lugar de aplicarlo en toda la imagen.

Si se compara el algoritmo de clasificación de regiones con texturas homogéneas basado en el ratio DC con el algoritmo basado en el cálculo de gradientes utilizado en [10] y [36], se observa según los experimentos realizados que la clasificación basada en el ratio DC es cerca de 4 veces más rápida en media ($\Delta\text{Time} \approx -75\%$). En la comparativa sólo se

contempla el cálculo del gradiente, el ángulo y la amplitud asociada. La operación *atan* que se requiere en tanto en [10] como en [36] fue optimizada conforme a las indicaciones de [36]. Además, en la comparativa no se contempla el cálculo de los requeridos histogramas ni otras operaciones, por lo que la mejora realmente es superior.

La integración realizada mejora los resultados obtenidos por [10], donde el tiempo de codificación es reducido 56.8% a expensas de un incremento en el tamaño del bitstream del 2.5% en media. En la Tabla 5.8 se indican los resultados tras la integración. Como se puede observar, el tiempo de ejecución se reduce hasta un 80% con un incremento en el tamaño del bitstream generado de 1.88% y sin pérdida de calidad apreciable. Los resultados que se muestran se corresponden a la media de los resultados obtenidos utilizando nueve valores de QP diferentes: 10, 15, 20, 25, 30, 35, 40, 45 y 50.

Tabla 5.8. Resultados tras la integración del algoritmo de selección de tamaño de bloque basado en el ratio DC con [10] para la configuración intra-only.

Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔEncoding Time (%)	Max ΔBitstream (%)
blue_sky 1080p25	-0.04	-0.09	-52.50	0.92
pedestrian_area 1080p25	0.43	-0.13	-72.18	1.84
riverbed 1080p25	-0.63	-0.06	-62.82	-0.05
rush_hour 1080p25	-0.01	-0.13	-80.08	1.88

5.6.1.1.2 Verificación en un codificador en tiempo real

Tras verificar el funcionamiento del algoritmo propuesto utilizando el software HM, se procedió a su implementación en un software que permita la codificación HEVC en tiempo real. Para tal propósito se utilizó el codificador x265, implementando en su código el algoritmo de decisión de tamaño de bloque basado en texturas homogéneas y el basado en gradientes definido en [10]. En los primeros experimentos se observó que el coste computacional asociado al algoritmo presentado en [10] es excesivo para permitir la codificación en tiempo real para formatos UHD, ya que el cómputo asociado a los cálculos de la amplitud y ángulo del gradiente y de los histogramas requeridos, es muy superior a la reducción en el tiempo de codificación que aporta el algoritmo propiamente dicho. El uso de una GPU podría ser una solución óptima para paliar este problema pero estas primeras pruebas estaban centradas exclusivamente en el uso de una CPU, por lo que la implementación en GPU se abordará en los próximos apartados.

Debido a esto, y con el objetivo de reafirmar que el algoritmo diseñado de decisión de tamaño de bloque puede coexistir con otras soluciones existentes, se decidió integrarlo con el algoritmo propuesto en [53] aplicándolo en aquellas regiones que no presentan texturas homogéneas. El trabajo presentado en [53] aplica un algoritmo de decisión de tamaño de bloque optimizado para CU tipo inter por medio de una consulta de los vecinos espaciales y de la CU co-situada. En [53] el proceso recursivo de división de la imagen en tamaños de bloque inferiores es detenido si se cumplen las siguientes condiciones:

- El valor de depth de la CU co-situada es más pequeño o igual al valor de depth que se está evaluando, y si son iguales, el mejor modo de partición inter para la depth que se está evaluando no debe de ser NxN.
- El valor de depth de dos o más CU vecinas es menor o igual al valor de depth que se está evaluando. En caso de ser iguales, para que se considere en el cómputo, el modo de partición inter para esa depth no debe de ser NxN.
- Que no se esté codificando una slice tipo I.

En la Tabla 5.9 se muestran las prestaciones alcanzadas por la integración realizada en el software x265 v1.7 utilizando el preset de configuración *ultrafast* para secuencias de formato UHD. En comparación con el software x265 sin ninguna modificación, las optimizaciones realizadas consiguen reducir hasta un 47% los tiempos de codificación con una pérdida inapreciable de calidad. La reducción en tiempos de ejecución en media es cercana a 23.4%, siendo la mayor pérdida de calidad visual 0.168 dB la cual es imperceptible a 20 Mbps.

Tabla 5.9. Resultados tras la integración del algoritmo de selección de tamaño de bloque basado en el ratio DC con el algoritmo propuesto [53] utilizando el codificador x265 en secuencias UHD.

Sequence	Bitrate	ΔPSNR (dB)	ΔEncoding Time (%)
crowd_run 3840x2160	10 Mbps	-0.026	-12.36
	15 Mbps	-0.078	-9.25
	20 Mbps	-0.097	-21.79
ducks_take_off 3840x2160	10 Mbps	-0.018	-30.61
	15 Mbps	-0.074	-27.89
	20Mbps	-0.168	-47.00
in_to_tree 3840x2160	10 Mbps	0.029	-16.87
	15 Mbps	0.087	-29.03
	20 Mbps	0.114	-30.99
old_town_cross 3840x2160	10 Mbps	0.006	-18.42
	15 Mbps	-0.013	-14.15
	20 Mbps	-0.082	-25.95
park_joy 3840x2160	10 Mbps	0.127	-25.85
	15 Mbps	0.041	-16.84
	20 Mbps	-0.093	-24.10

5.6.1.1.3 Verificación en aplicaciones médicas

La reducción en la tasa de datos generada que proporciona el estándar HEVC respecto al H.264 puede resultar considerablemente beneficiosa para aplicaciones utilizadas en entornos médicos como puede ser la telemedicina o el almacenamiento de las imágenes resultantes de una resonancia magnética o una tomografía axial computarizada (TAC).
Las imágenes utilizadas dentro de este contexto presentan gran cantidad de regiones con texturas homogéneas por lo que los algoritmos que reducen la carga computacional en este tipo de áreas resultan sumamente útiles. En el trabajo presentado en [51] se demuestra cómo el algoritmo de decisión de tamaño de bloque basado en el análisis de la imagen en busca de regiones con texturas homogéneas reduce los tiempos de ejecución de la codificación HEVC sin afectar a la calidad visual de las imágenes médicas codificadas.

Existe un conjunto de extensiones dentro del estándar HEVC denominadas *Range Extensions (RExt)* [54] destinadas a determinadas aplicaciones de vídeo, siendo una de estas aplicaciones los entornos médicos. La codificación sin pérdidas se utiliza en el entorno médico para proporcionar imágenes con alto grado de detalle que faciliten la realización de un diagnóstico certero. HEVC mejora considerablemente las prestaciones del modo de codificación sin pérdidas respecto a sus predecesores y a otros estándares como JPEG-2000 o JPEG-LS [55]. En el caso de la codificación utilizando únicamente planos tipo I, modo de codificación que también se utiliza para proporcionar secuencias de vídeo de alta calidad, HEVC también ofrece mejores prestaciones que JPEG, JPEG2000, JPEG XR o H.264.

En el modo de codificación sin pérdidas de HEVC se eliminan las operaciones asociadas a la transformación, cuantificación y los filtrados SAO y deblocking. De esta forma se proporciona una gran fidelidad entre la imagen codificada y la original, al mismo tiempo que se consigue una reducción en la tasa de datos generada que oscila entre el 3.2% y 13.2% respecto a la imagen sin comprimir. El esquema asociado a la codificación sin pérdidas se muestra en la Figura 5.25.

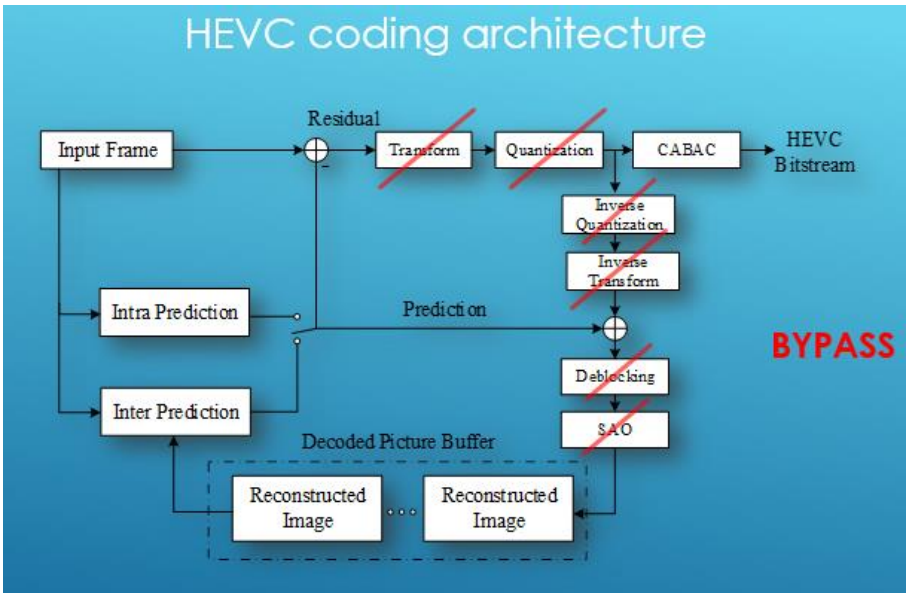


Figura 5.25. Esquema de codificación sin pérdidas en HEVC.

En el modo de codificación sin pérdidas se utiliza la codificación DPCM aplicada horizontal y verticalmente a cada muestra de las que componen el residuo. Esta técnica conocida *Residual Differential Pulse Code Modulation (RDPCM)*, permite la reducción de la tasa de datos generada explotando la redundancia existente entre residuos [54]. El modo de codificación sin pérdidas de HEVC incluye tanto predicción inter como intra, por lo que las optimizaciones propuestas pueden utilizarse también en este modo de codificación.

Para verificar el funcionamiento del algoritmo de decisión de tamaño de bloque basado en la clasificación de regiones con texturas homogéneas se ha utilizado un conjunto de secuencias de vídeo pertenecientes a *The Cancer Imaging Archive (TCIA)* [56]. TCIA ofrece gran cantidad de imágenes médicas de acceso público. En la Tabla 5.10 se muestran las secuencias de vídeo utilizadas para la verificación del algoritmo, y en la Figura 5.26 se muestra un plano representativo de cada secuencia de vídeo empleada.

Las imágenes tienen el formato de fichero DICOM por lo que requieren ser convertidas a YUV para poder ser codificadas empleando el software HM. Para realizar esta conversión

las imágenes fueron convertidas del formato DICOM a AVI mediante el software *MicroDicom* [57] para ser posteriormente convertidas a YUV 4:2:0 mediante *FFmpeg* [58]. En entornos médicos se utiliza el perfil monocromático de HEVC. Por lo que antes de proceder a la codificación de las secuencias de vídeo se eliminaron las componentes de la crominancia, para obtener finalmente secuencias YUV 4:0:0.

Tabla 5.10. Descripción de las secuencias de vídeo para diagnóstico médico utilizadas.

Collection	**Subject ID**	**Study Instance ID**	**Series ID**	**Pseudonym**
REMBRANDT	900-00-1961	…04452	... 09108	Brain 256x256
CT COLONOGRAPHY	1.3.6.1.4.1.9328.50.4.0001	…8.50.4.1	…0112	CTColonography 512x512
NaF-PROSTATE	NaF-PROSTATE-01-0001	…6504	…90112	Prostate 512x512
BREAST-DIAGNOSIS	BreastDx-01-0001	…44317	…75663	T2W-SENSE 640x640
TCGA-LIHC	TCGA-BC-4073	…85289	…56942	TCGA-LIHC 320x320
TCGA-OV	TCGA-09-0367	…1897360	…75499	TCGA-OV 512x512
TCGA-STAD	TCGA-VQ-A8DL	…20505	…00977	TCGA-STAD 512x512
TCGA-THCA	TCGA-DE-A4MA	..7769	..0599	TCGA-THCA 512x512
TCGA-UCEC	TCGA-D1-A0ZO	..1274	..3457	TCGA-UCEC 512x512

Las Tablas 5.11, 5.12 y 5.13 muestran los resultados obtenidos aplicando el algoritmo optimizado de decisión de tamaño de bloque en entornos médicos utilizando el perfil monocromático del estándar HEVC en el software HM. Al igual que en tablas similares mostradas en apartados anteriores, los resultados que se muestran se corresponden a la media de los resultados obtenidos utilizando nueve valores de QP diferentes: 10, 15, 20, 25, 30, 35, 40, 45 y 50. Como puede observarse en estas tablas se ha obtenido una considerable reducción en los tiempos de ejecución, alcanzando una reducción de hasta un 57.9% en el mejor de los casos. El tamaño del bitstream generado es reducido para varias secuencias, siendo 0.352% la mayor reducción, y en general el incremento en el resto de secuencias es muy pequeño, siendo 0.24% el mayor incremento. La reducción en la calidad visual obtenida es imperceptible, siendo -0.09 dB el peor caso.

En las Tablas 5.14, 5.15 y 5.16 se pueden observar los resultados obtenidos bajo las mismas condiciones de test empleando el modo de codificación sin pérdidas. Para este modo, la reducción en los tiempos de ejecución oscila entre 8.16% y 76.5%, dependiendo directamente del número de bloques que presenten texturas homogéneas dentro de la secuencia de vídeo. El incremento en el tamaño del bitstream permanece siendo muy bajo, tomando el valor de 0.651% en el peor de los casos.

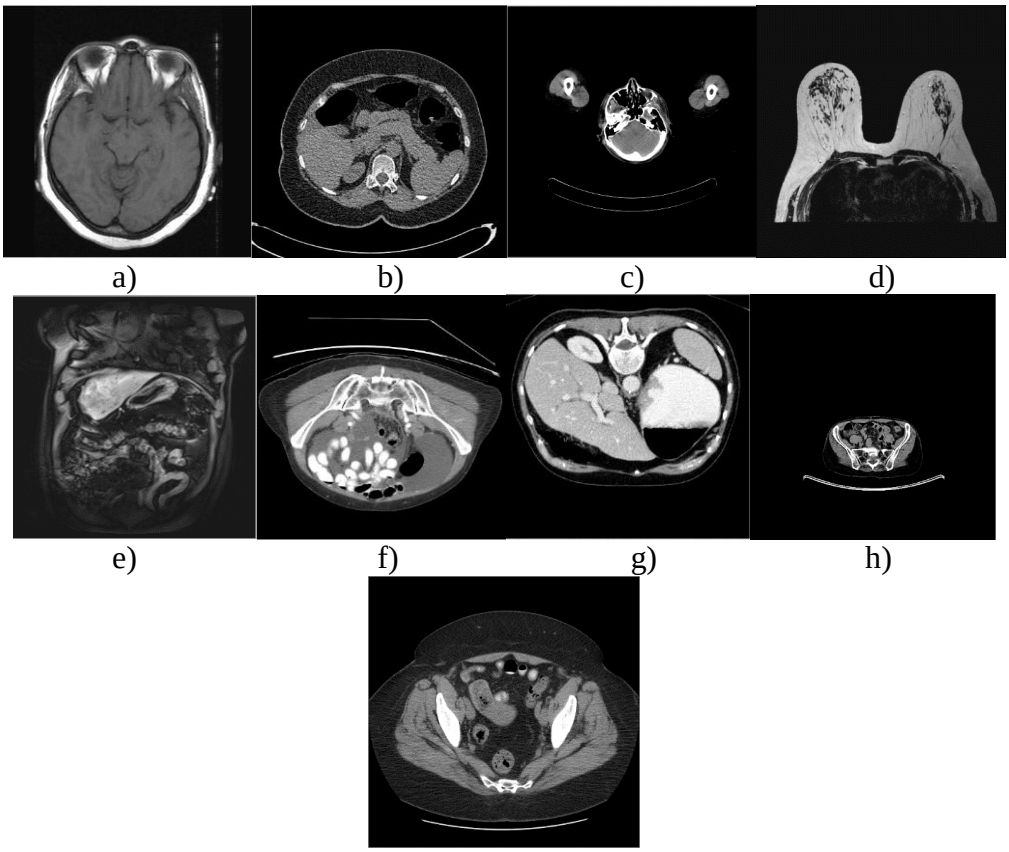


Figura 5.26. Planos representativos de las secuencias de vídeo utilizadas. a) Brain 256x256 b) CTColonography 512x512 c) Prostate 512x512 d) T2W-SENSE 640x640 e) TCGA-LIHC 320x320 f) TCGA-OV 512x512 g) TCGA-STAD512x512 h) TCGA-THCA512x512 i) TCGA-UCEC512x512.

Tabla 5.11. Resultados obtenidos para el perfil monocromático utilizando la configuración intra-only.

Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔEncoding Time (%)
Brain 256x256	-0.30	-0.06	-36.37
CTColonography 512x512	0.02	-0.01	-25.32
Prostate 512x512	0.07	-0.05	-52.84
T2W-SENSE 640x640	0.002	-0.01	-52.05
TCGA-LIHC 320x320	-0.16	-0.09	-39.83
TCGA-OV 512x512	0.24	-0.05	-35.87
TCGA-STAD 512x512	0.24	-0.05	-35.87
TCGA-THCA 512x512	0.24	-0.05	-35.87
TCGA-UCEC 512x512	0.11	-0.04	-30.86

Tabla 5.12. Resultados obtenidos para el perfil monocromático utilizando la configuración low-delay P.

Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔEncoding Time (%)	ΔIntra Prediction Time (%)	ΔInter Prediction Time (%)
Brain 256x256	-0.352	-0.037	-17.396	-28.384	-14.630
CTColonography 512x512	-0.015	-0.004	-13.429	-7.483	-14.924
Prostate 512x512	0.023	-0.008	-33.427	-14.908	-36.230
T2W-SENSE 640x640	0.029	-0.002	-28.458	-22.242	-29.475
TCGA-LIHC 320x320	-0.028	-0.009	-6.945	-17.037	-4.597
TCGA-OV 512x512	0.026	-0.027	-19.211	-20.161	-18.967
TCGA-STAD 512x512	0.044	-0.016	-17.111	-17.744	-16.963
TCGA-THCA 512x512	-0.005	0.004	-55.629	-20.434	-58.870
TCGA-UCEC 512x512	-0.019	-0.013	-15.166	-17.191	-14.814

Tabla 5.13. Resultados obtenidos para el perfil monocromático utilizando la configuración random-access.

Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔEncoding Time (%)	ΔIntra Prediction Time (%)	ΔInter Prediction Time (%)
Brain 256x256	-0.214	-0.024	-17.172	-28.921	-14.696
CTColonography 512x512	-0.007	-0.003	-14.454	-12.778	-14.844
Prostate 512x512	0.021	-0.009	-36.646	-21.096	-38.839
T2W-SENSE 640x640	0.092	-0.004	-30.481	-24.773	-31.330
TCGA-LIHC 320x320	0.063	-0.009	-8.292	-18.760	-6.232
TCGA-OV 512x512	-0.044	-0.030	-20.775	-21.407	-20.675
TCGA-STAD 512x512	0.064	-0.018	-18.934	-19.623	-18.828
TCGA-THCA 512x512	0.004	-0.004	-57.895	-30.767	-60.207
TCGA-UCEC 512x512	0.067	-0.014	-15.511	-17.180	-15.363

Tabla 5.14. Resultados obtenidos en el modo de codificación sin pérdidas utilizando la configuración intra-only.

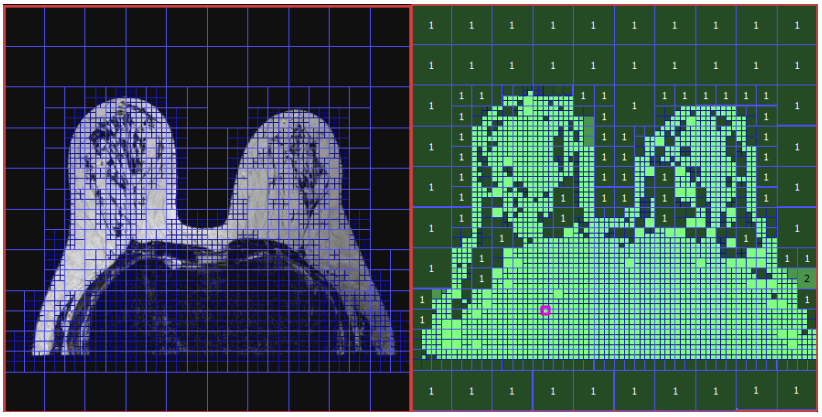
Sequence	ΔBitstream (%)	ΔEncoding Time (%)
Brain 256x256	0.651	-46.006
CTColonography 512x512	-0.013	-35.625
Prostate 512x512	0.136	-62.600
T2W-SENSE 640x640	0.216	-56.843
TCGA-LIHC 320x320	-0.014	-25.875
TCGA-OV 512x512	0.080	-48.762
TCGA-STAD 512x512	0.095	-38.747
TCGA-THCA 512x512	0.345	-76.511
TCGA-UCEC 512x512	0.040	-35.904

Tabla 5.15. Resultados obtenidos en el modo de codificación sin pérdidas utilizando la configuración low-delay P.

Sequence	ΔBitstream (%)	ΔEncoding Time (%)	ΔIntra Prediction Time (%)	ΔInter Prediction Time (%)
Brain 256x256	0.338	-20.251	-43.238	-16.891
CTColonography 512x512	0.008	-14.616	-14.343	-14.753
Prostate 512x512	-0.004	-37.501	-17.293	-38.686
T2W-SENSE 640x640	0.136	-34.713	-30.887	-35.148
TCGA-LIHC 320x320	0.093	-8.166	-24.682	-5.861
TCGA-OV 512x512	0.013	-23.692	-27.458	-23.399
TCGA-STAD 512x512	0.098	-18.793	-19.884	-18.750
TCGA-THCA 512x512	0.010	-59.072	-19.111	-60.787
TCGA-UCEC 512x512	0.017	-14.691	-17.505	-14.642

Tabla 5.16. Resultados obtenidos en el modo de codificación sin pérdidas utilizando la configuración random-access.

Sequence	ΔBitstream (%)	ΔEncoding Time (%)	ΔIntra Prediction Time (%)	ΔInter Prediction Time (%)
Brain 256x256	0.323	-21.101	-43.004	-17.336
CTColonography 512x512	0.012	-14.214	-14.689	-14.287
Prostate 512x512	0.003	-36.532	-19.951	-37.881
T2W-SENSE 640x640	0.144	-33.351	-32.105	-33.589
TCGA-LIHC 320x320	0.088	-9.329	-25.046	-6.633
TCGA-OV 512x512	0.026	-22.850	-27.670	-22.305
TCGA-STAD 512x512	0.073	-17.963	-20.998	-17.609
TCGA-THCA 512x512	0.017	-57.562	-26.600	-59.576
TCGA-UCEC 512x512	0.018	-14.858	-18.367	-14.472



a) b)

Figura 5.27. Resultado de la clasificación de texturas propuesta. a) Resultado del algoritmo de decisión de tamaño de bloque. b) En verde oscuro se muestran los bloques que presentan texturas homogéneas.

En la Figura 5.27 se muestran los tamaños de bloque seleccionados por el codificador HEVC aplicando el algoritmo optimizado. En la imagen de la derecha de la Figura 5.27, los bloques que aparecen en verde oscuro indican los bloques que han sido clasificados como bloques que presentan texturas homogéneas.

5.6.1.2 **Selección del modo de predicción intra**

En el trabajo presentado en [47] se muestran los resultados obtenidos para todos algoritmos de reducción de complejidad del estándar HEVC basados en la clasificación de regiones que contienen texturas homogéneas que se describen en la presente Tesis. En el caso de la optimización del algoritmo de selección del mejor modo de predicción intra, si la CU en cuestión presenta texturas homogéneas, entonces se reduce el número de modos a evaluar de los 35 que permite el estándar HEVC a únicamente 4 o 5 modos. Los modos de predicción que conforman las lista de candidatos a evaluar son el modo DC, Plano, Horizontal, Vertical y el modo MPM, en el caso de no ser uno de los anteriores. Adicionalmente, la función de coste basada en RDO para la evaluación de los distintos candidatos es eliminada, utilizando únicamente costes basados en SATD. En la Tabla 5.17 se muestra la reducción en tiempos de ejecución, el incremento en el tamaño del bitstream y la variación de la calidad visual obtenidos por el algoritmo optimizado de selección de modo de predicción intra integrado junto al algoritmo optimizado de decisión de tamaño de bloque implementados en el software HM v16.2. Para reducir el tiempo de ejecución de las pruebas, los resultados que se presentan en la Tabla 5.17 han sido obtenidos utilizando un tamaño máximo de CTU de 32x32 e igualando el tamaño de la TU siempre al tamaño de la PU (parámetro de configuración $TU_{MaxDepth} = 1$).
Los resultados que se muestran se corresponden a la media de los resultados obtenidos utilizando nueve valores de QP diferentes: 10, 15, 20, 25, 30, 35, 40, 45 y 50.

Tabla 5.17. Prestaciones obtenidas por el algoritmo de decisión de tamaño de bloque y el algoritmo de decisión de modo de predicción intra para la configuración intra-only.

Sequence	Intra Block Size Decision			
	ΔBitstream (%)	**ΔPSNR (dB)**	**ΔTime (%)**	**ΔIntra Prediction Time (%)**
blue_sky 1080p25	-0.144	-0.045	-40.61	-42.19
pedestrian_area 1080p25	-0.015	-0.052	-41.47	-42.98
riverbed 1080p25	-0.113	-0.014	-12.68	-13.43
rush_hour 1080p25	-0.531	-0.052	-46.25	-47.93
shields 720p59	-0.068	-0.015	-6.67	-6.99
stockholm 720p60	-0.089	-0.026	-17.33	-17.80

Sequence	Intra Block Size Decision + Intra Mode Decision			
	ΔBitstream (%)	**ΔPSNR (dB)**	**ΔTime (%)**	**ΔIntra Prediction Time (%)**
blue_sky 1080p25	-0.067	-0.074	-44.55	-47.07
pedestrian_area 1080p25	0.371	-0.093	-47.76	-51.05
riverbed 1080p25	-0.297	-0.037	-20.35	-24.39
rush_hour 1080p25	-0.216	-0.096	-54.77	-59.15
shields 720p59	-0.098	-0.038	-9.32	-9.85
stockholm 720p60	-0.194	-0.052	-20.85	-21.71

5.6.1.3 Predicción inter

En el conjunto de experimentos cuyos resultados se muestran en este apartado, se pretende evaluar el algoritmo optimizado encargado de realizar la predicción inter. Los test fueron realizados utilizando la configuración low-delay P para incluir slices tipo inter que contengan CU intra e inter. Para estas pruebas se incluyeron nuevas secuencias de vídeo que contienen diferentes tipos de movimiento. En la Tabla 5.18 se muestran los resultados obtenidos combinando el algoritmo de decisión de tamaño de bloque con el algoritmo de decisión de modo de partición inter utilizando la configuración low-delay P. Al igual que para la Tabla 5.17, para reducir el tiempo de ejecución de las pruebas, los resultados que se presentan han sido obtenidos utilizando un tamaño máximo de CTU de 32x32 e igualando el tamaño de la TU siempre al tamaño de la PU. Como en tablas anteriores, se muestra la media de las pruebas realizadas utilizando nueve valores de QP diferentes: 10, 15, 20, 25, 30, 35, 40, 45 y 50; implementando el algoritmo propuesto en el software HM v16.2.

5.6.1.4 Verificación del flujo de codificación completo basado en la detección de texturas homogéneas

En este apartado se procederá a verificar las prestaciones de todos los algoritmos de optimización que se han implementado que están basados en la detección de regiones que presentan texturas homogéneas conforme al flujo de codificación indicado en la Figura 5.17. En este apartado no se incluye el filtrado de la imagen de entrada, ya que se trata de una mejora de calidad visual subjetiva y las métricas que se emplean para caracterizar el algoritmo y realizar una posterior comparativa con trabajos ya existentes se basan en métricas objetivas (PSNR y BD-Rate), pudiendo llevar a una mala interpretación de los resultados obtenidos. Por el mismo motivo, no se contempla la variación de QP en regiones con texturas homogéneas. Como ya se comentó previamente, la verificación se realiza con el algoritmo RC deshabilitado para independizar la codificación de la selección de QP que realice el RC, analizando para un determinado nivel de calidad prefijado por el valor de QP, cómo varían las prestaciones del codificador. Por otro lado, el RC no es un algoritmo estandarizado por lo que diferentes implementaciones podrían llevar a resultados muy

dispares. Las optimizaciones deshabilitadas son evaluadas independientemente en apartados posteriores.

Tabla 5.18. Prestaciones obtenidas por el algoritmo de decisión de tamaño de bloque y el algoritmo de decisión de modo de partición inter para la configuración low-delay P.

Sequence	ΔBits (%)	ΔPSNR (dB)	ΔTime (%)	ΔIntra Prediction Time (%)	ΔInter Prediction Time (%)
blue_sky 1080p25	0.13	-0.047	-37.02	-21.17	-38.57
pedestrian 1080p25	0.549	-0.04	-29.82	-32.2	-28.01
riverbed 1080p25	-0.01	-0.013	-5.98	-19.13	-2.15
rush_hour 1080p25	-0.09	-0.031	-34.51	-40.21	-32.96
parkjoy 1080p25	0.013	-0.022	-1.64	-2.89	-0.87
sunflower 1080p25	-0.16	-0.036	-21.3	-28.21	-19.39
tractor 1080p25	0.147	-0.021	-9.6	-16.03	-8.08
stockholm 720p60	-0.32	-0.013	-18.11	-28.98	-17.21
shields 720p59	0.018	-0.016	-5.01	-19.93	-4.01

Respecto a las pruebas realizadas en apartados anteriores, para la verificación del flujo total se han añadido varias secuencias de resolución 4K, ya que una de las principales ventajas que aporta HEVC respecto a sus predecesores es permitir la codificación de esta resolución generando una tasa de datos admisible para su posible transmisión. El rango de QP utilizado para las secuencias 4K es reducido a únicamente 10, 15, 20, 25, 30 y 35 debido a que las aplicaciones que utilizan resolución 4K requieren gran calidad visual por lo que carece de sentido utilizar valores de QP elevados. Para las resoluciones 1080p y 720p, se siguen manteniendo los mismos nueve valores de QP que en pruebas anteriores (10, 15, 20, 25, 30, 35, 40, 45 y 50).

En la Tabla 5.19 se muestran los resultados de estas pruebas para las tres configuraciones utilizadas en el software HM v16.2. Como se puede apreciar la reducción en tiempos de ejecución puede llegar a ser de hasta 77.92% en el mejor de los casos. En general, el tamaño del bitstream también es reducido, siendo 2.2% el mayor decremento y 0.5% el mayor incremento. La calidad visual no se ve excesivamente perjudicada, siendo 0.34 dB la mayor pérdida. Por lo tanto, los algoritmos propuestos demuestran tener una gran eficiencia.

Tabla 5.19. Prestaciones obtenidas por los algoritmos de optimización basados en la clasificación de regiones con texturas homogéneas.

Sequence	All Intra			Low Delay P			Random Access		
	ΔBits (%)	ΔPSNR (dB)	ΔTime (%)	ΔBits (%)	ΔPSNR (dB)	ΔTime (%)	ΔBits (%)	ΔPSNR (dB)	ΔTime (%)
Bosphorous 4K	-0.67	-0.32	-77.23	-1.64	-0.15	-77.92	-2.20	-0.14	-77.92
Coastguard 4K	0.24	-0.05	-57.6	0.38	-0.04	-59.76	0.50	-0.04	-59.76
Jockey 4K	-2.11	-0.34	-67.09	-1.58	-0.16	-68.22	-2.20	-0.14	-68.22
ShakeNDry 4K	-1.21	-0.2	-35.78	-0.89	-0.1	-43.40	-1.14	-0.1	-43.40
Blue_Sky 1080p25	-0.08	-0.1	-43.05	-0.37	-0.07	-57.11	-0.63	-0.1	-57.11
Pedestrian_Area 1080p25	0.48	-0.12	-46.64	0.12	-0.05	-32.93	0.03	0.06	-49.88
Riverbed 1080p25	-0.34	-0.04	-21.85	-0.11	-0.03	-23.57	-0.21	-0.02	-36.70
RushHour 1080p25	-0.07	-0.1	-52.17	-0.31	-0.05	-38.24	-0.41	-0.05	-50.75
Aspen 1080p30	-0.51	-0.18	-60.88	-0.52	-0.08	-46.26	-0.63	-0.07	-61.80
CrowdRun 1080p50	-0.22	-0.07	-20.15	-0.43	-0.07	-26.48	-0.36	-0.07	-40.75
DucksTakeOff 1080p50	-0.29	-0.06	-14.88	-0.11	-0.04	-21.73	-0.20	-0.04	-7.84
InToTree 1080p50	-0.79	-0.14	-35.06	-1.22	-0.07	-37.29	-1.43	-0.07	-35.08
Life 1080p30	0.31	-0.16	-33.49	-0.03	-0.1	-26.04	0.04	-0.13	-52.06
ParkJoy 1080p25	-0.25	-0.03	-8.65	-0.56	-0.05	-21.38	-0.04	-0.05	-40.78
RedKayak 1088p25	-0.09	-0.14	-50.74	0.335	-0.14	-39.49	0.37	-0.09	-34.83
SnowMountain 1080p30	-0.002	-0.17	-37.19	-0.83	-0.1	-39.38	-0.40	-0.1	-54.63
Sunflower 1080p25	-0.16	-0.08	-35.63	-0.41	-0.04	-25.36	-0.83	-0.05	-53.29
Tractor 1080p25	-0.04	-0.06	-23.68	-0.3	0.005	-22.53	-0.44	-0.06	-39.97
MobileCalendar 720p50	-0.14	-0.04	-14.88	-0.37	-0.05	-20.25	-0.70	-0.06	-36.30
ParkRun 720p50	-0.02	-0.04	-10.77	-0.25	-0.06	-24.58	0.40	-0.04	-34.42
Shields 720p59	-0.32	-0.05	-12.88	-0.27	-0.06	-19.40	-0.72	-0.06	-35.95
Stockholm 720p60	-0.09	-0.06	-23.65	-0.54	-0.06	-28.91	-0.56	-0.06	-35.97

5.6.1.5 Comparativa con el Estado del Arte.

En este apartado se realizará una comparativa entre el flujo de codificación propuesto y los trabajos más relevantes encontrados en la literatura existente, en términos de BD-rate e incremento en el tiempo de ejecución. Para tal propósito se han utilizado las condiciones de test sugeridas por JCT-VT [33]. En concordancia con dichas recomendaciones sólo se han utilizado los valores de QP = 22, 27, 32 y 37, presentando en las tablas comparativas una media de los resultados obtenidos para estos cuatro valores.

En la Tabla 5.20 se muestra un análisis detallado de los resultados obtenidos para cada secuencia empleada utilizando diferentes configuraciones. Como se puede observar, el flujo propuesto obtiene una considerable reducción en cuanto a tiempo de ejecución se refiere respecto al software HM v16.2 sin ninguna modificación.

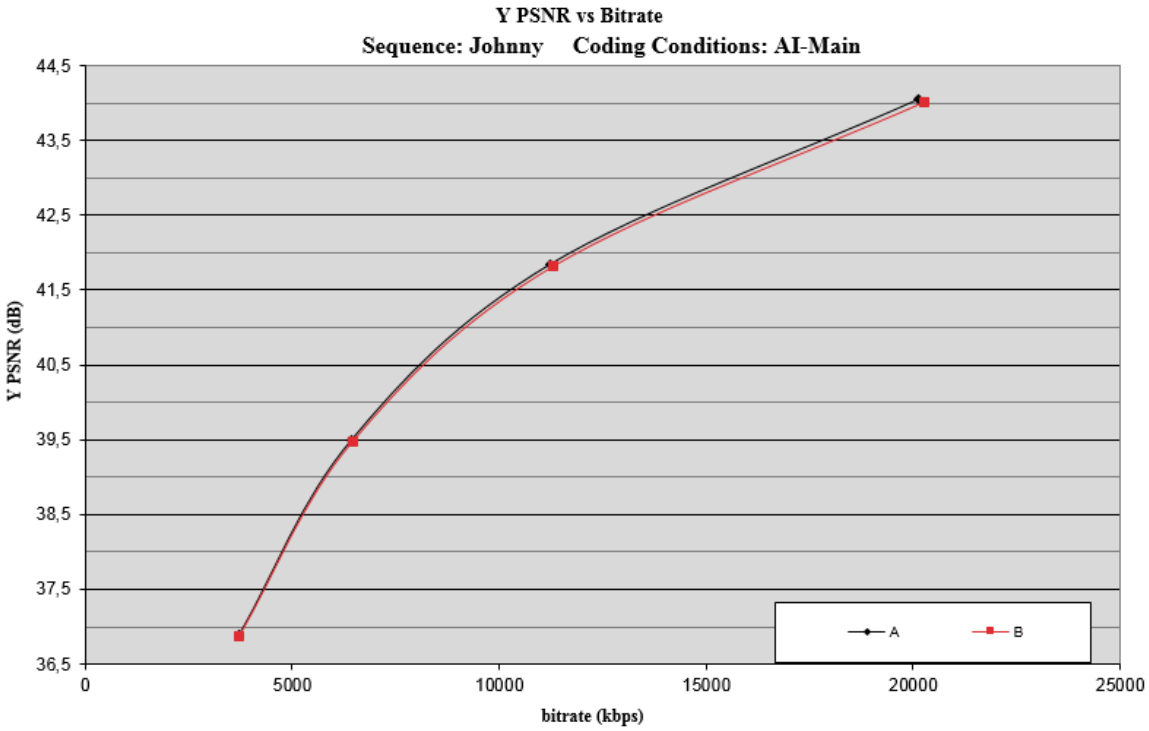
En la Figura 5.28 se muestra una comparativa entre el flujo de codificación propuesto y el software HM v16.2 sin modificar en términos de PSNR/bitrate, para las secuencias *Johnny* and *BasketballDrive*, al ser las secuencias en las que se consigue mayor reducción en los tiempos de ejecución, comprobando que esta reducción no conlleva un incremento en el tamaño de bitstream excesivo ni una pérdida de calidad visual perceptible.

En las Tablas 5.21 y 5.22 se establece la comparativa con otros trabajos de gran relevancia para la configuración intra-only (también llamada *all intra*). Como se puede apreciar el flujo de codificación propuesto supera tanto en términos de BD-rate como en ΔT a los trabajos presentados en [36, 14-16].

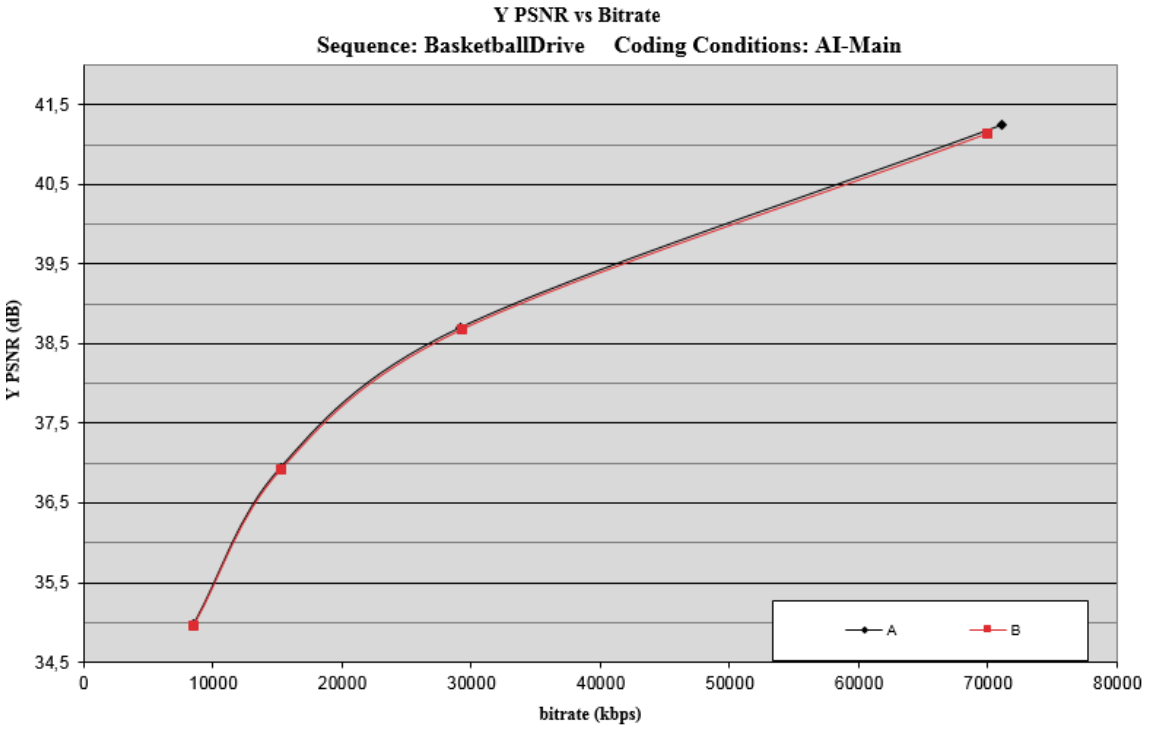
En las Tablas 5.23 y 5.24 se muestran las comparativas realizadas para las configuraciones low-delay P y random-access. En estas tablas se observa que los trabajos [7] y [18] son los que proporcionan resultados similares a la propuesta realizada en términos de calidad, sin embargo el flujo de codificación propuesto alcanza mayores reducciones en tiempos de ejecución. Para establecer qué trabajos pueden considerarse como los óptimos en términos de eficiencia en la reducción de la complejidad se han utilizado curvas de Pareto. Los mejores métodos pueden ser identificados como aquellos pertenecientes al frente de Pareto en el espacio BD-rate/Reducción en tiempos de ejecución. En las Figuras 5.29, 5.30 y 5.31 se muestras las gráficas en las que se indica el frente de Pareto (en verde) para las diferentes configuraciones utilizadas. Como se observa en estas figuras, el método de codificación propuesto pertenece al frente de Pareto en todos los casos, demostrando la eficiencia del flujo propuesto.

Tabla 5.20. Resultados finales para establecer comparativas utilizando las secuencias JCT-VT.

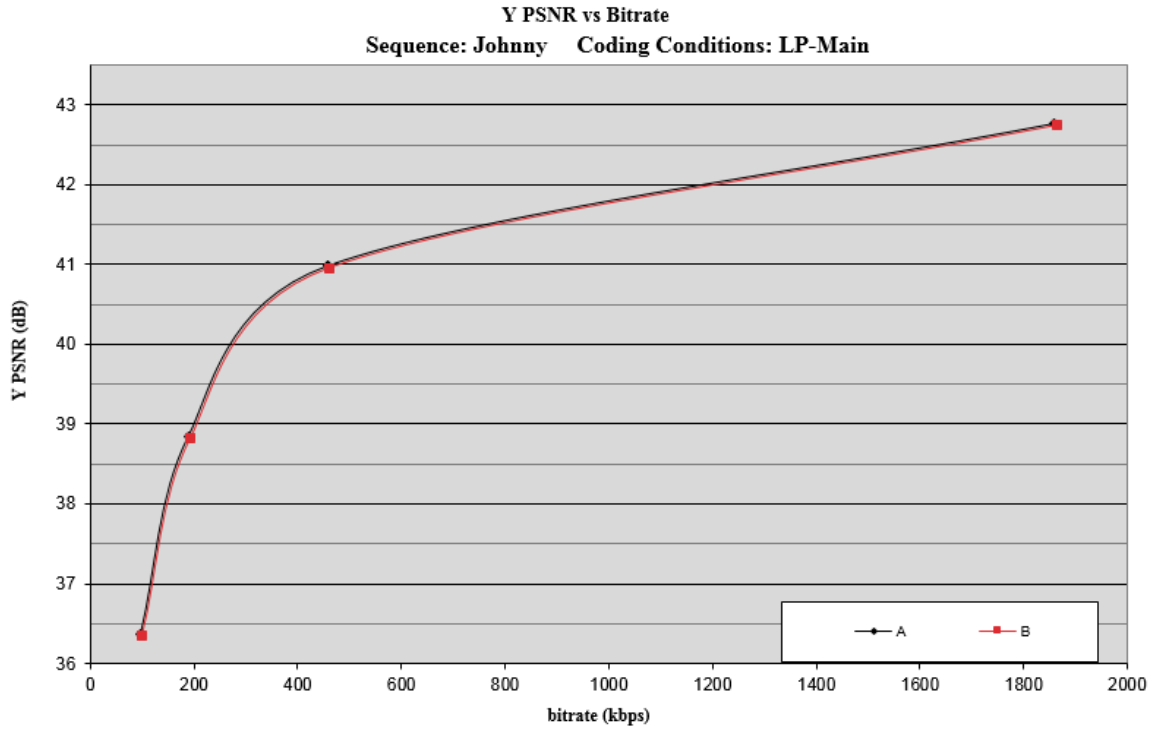
Class	Sequences	All Intra			Low Delay P			Random Access		
		BD-Rate	ΔT	BD-PSNR	BD-Rate	ΔT	BD-PSNR	BD-Rate	ΔT	BD-PSNR
A	Traffic	0.4	-39.46	-0.032	0.90	-27.49	-0.054	1.00	-27.81	-0.049
	PeopleOnStreet	0.4	-43.15	-0.025	1.30	-26.09	-0.114	1.10	-25.63	-0.079
B	Kimono	0.3	-30.82	-0.027	0.60	-22.76	-0.045	1.10	-20.57	-0.082
	ParkScene	0.1	-29.93	-0.003	1.10	-22.16	-0.116	0.80	-20.82	-0.074
	Cactus	1.4	-44.93	-0.068	1.20	-32.84	-0.043	1.50	-32.70	0.039
	BasketballDrive	1.4	-60.86	0.039	1.70	-46.40	-0.046	1.90	-46.92	0.040
	BQTerrace	0.9	-44.94	-0.045	1.10	-35.93	0.045	1.20	-35.47	0.083
C	BasketBallDrill	1.5	-46.12	-0.171	1.60	-32.08	-0.143	2.10	-32.33	-0.145
	BQMall	0.4	-35.2	-0.036	0.40	-23.61	-0.036	0.80	-24.87	-0.049
	PartyScene	0.1	-25.39	-0.010	0.20	-14.45	-0.013	0.50	-14.22	-0.023
	RaceHorses	0.1	-27.41	-0.004	0.50	-16.60	-0.050	0.50	-16.76	-0.042
D	BasketballPass	0.6	-45.46	-0.061	1.20	-29.17	-0.142	1.40	-30.49	-0.124
	BQSquare	0.4	-28.63	-0.037	0.50	-19.22	-0.025	0.70	-20.43	-0.032
	BlowingBubbles	0.1	-18.16	-0.009	0.10	-10.21	-0.001	0.30	-10.92	-0.013
	RaceHorses	0.1	-37.83	-0.008	0.50	-23.80	-0.052	0.60	-23.66	-0.039
E	FourPeople	0.5	-44.47	-0.048	1.00	-34.66	-0.065	1.10	-35.27	-0.072
	Johnny	1.2	-61.61	-0.105	1.90	-55.75	-0.083	2.30	-57.36	-0.131
	KristenAndSara	0.9	-53.22	-0.079	1.10	-44.56	-0.043	1.60	-46.41	-0.093
Average		0.58	-39.87	-0.041	0.94	-28.77	-0.057	1.14	-29.04	-0.049


Y PSNR vs Bitrate
Sequence: Johnny Coding Conditions: AI-Main
Y PSNR (dB)
44,5
43,5
42,5
41,5
40,5
39,5
38,5
37,5
36,5
0
5000
10000
15000
20000
25000
bitrate (kbps)
A
B

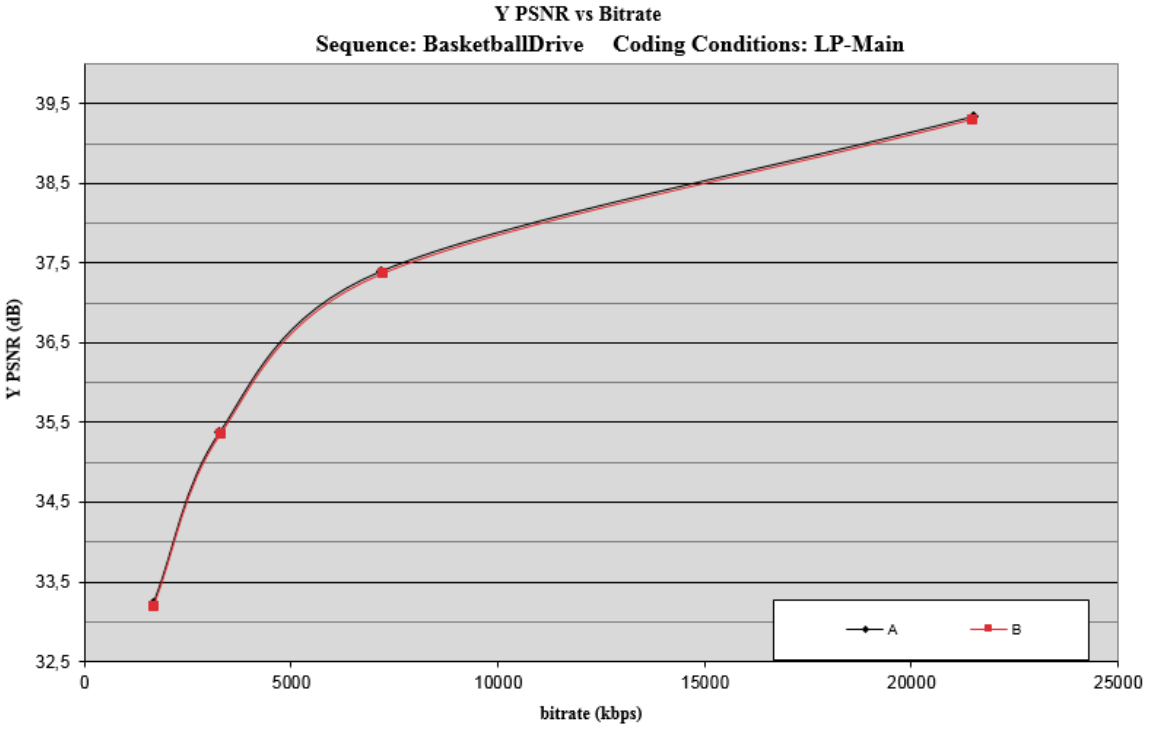
a)


Y PSNR vs Bitrate
Sequence: BasketballDrive Coding Conditions: AI-Main
Y PSNR (dB)
41,5
40,5
39,5
38,5
37,5
36,5
35,5
34,5
0
10000
20000
30000
40000
50000
60000
70000
80000
bitrate (kbps)
A
B

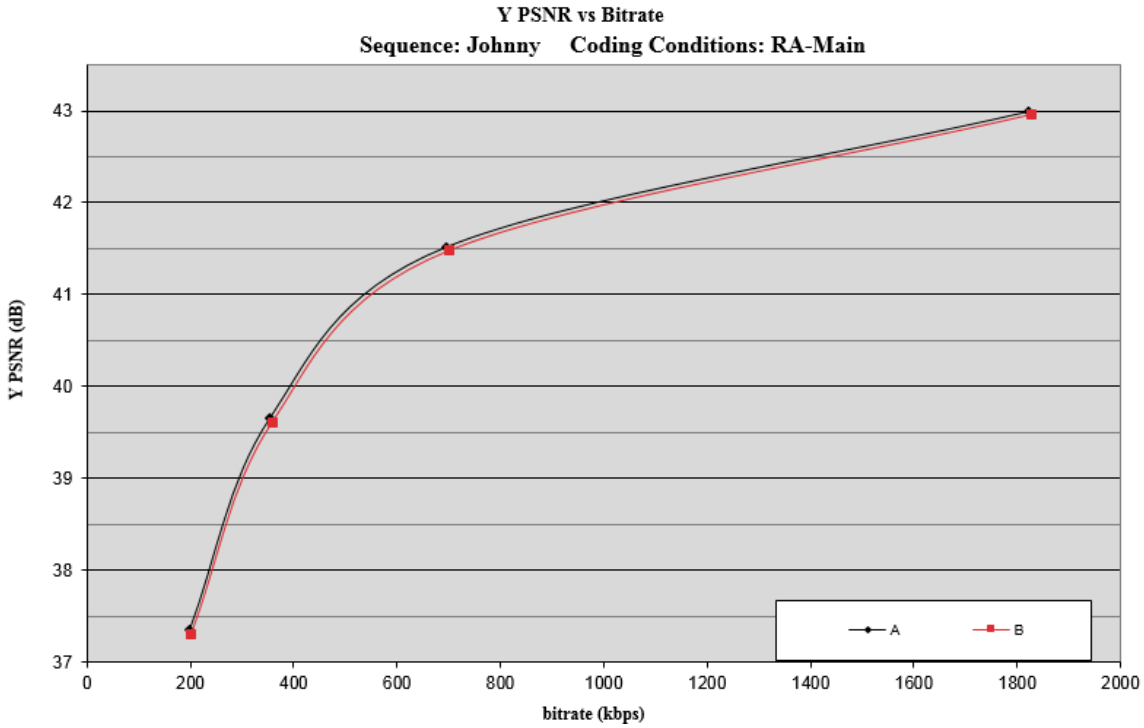
b)


Y PSNR vs Bitrate
Sequence: Johnny Coding Conditions: LP-Main
Y PSNR (dB)
43
42
41
40
39
38
37
36
0
200
400
600
800
1000
1200
1400
1600
1800
2000
bitrate (kbps)
A
B

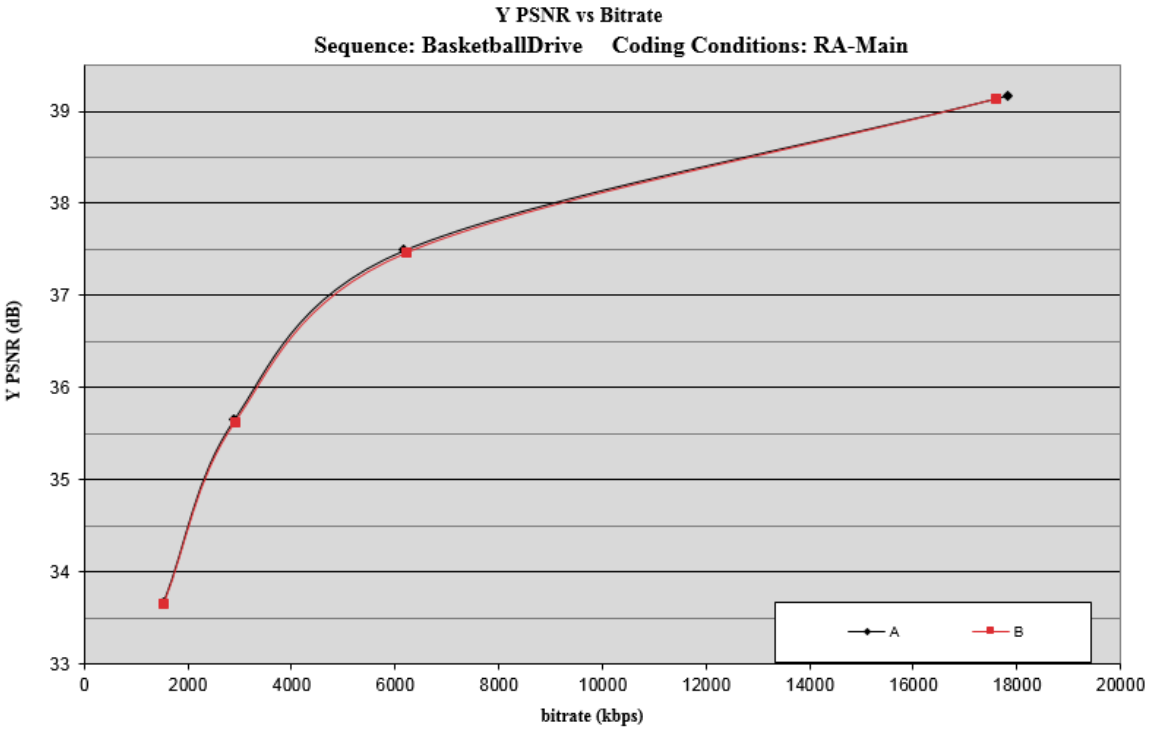
c)


Y PSNR vs Bitrate
Sequence: BasketballDrive Coding Conditions: LP-Main
Y PSNR (dB)
39,5
38,5
37,5
36,5
35,5
34,5
33,5
32,5
0
5000
10000
15000
20000
25000
bitrate (kbps)
A
B

d)



e)



f)

Figura 5.28. Comparación entre HMv16.2 (A) y la propuesta (B) en términos de Y-PSNR/Bitrate.
Secuencias Johny y BasketballDrive, utilizando las configuraciones intra-only, low-delay P y random-access.

Tabla 5.21. Comparativa con otros métodos existentes para la configuración intra-only.

Seq.	[8]		[36]		[6]		[16]		[15] α=0.7		[14]		[18]		[7]		Proposal	
	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT
Class A	0.85	-54.8	0.55	-20.4	1.14	-50	0.39	-37.8	1.41	-35	2.28	-21.9	1.15	-59	0.80	-56.2	0.40	-41.3
Class B	0.69	-58.7	0.70	-20.5	1.26	-53.2	0.56	-40.6	0.79	-24.3	2.16	-28.6	0.94	-61	0.72	-59.4	0.82	-42.3
Class C	0.80	-47.2	0.74	-19.5	0.60	-34.2	0.84	-37.3	1.85	-33.6	1.50	-17.8	1.08	-56.5	1.18	-50.7	0.53	-33.5
Class D	1.37	-46.9	0.94	-19.5	0.25	-34.5	0.84	-35.5	1.86	-31.1	1.19	-14.5	1.00	-54.7	1.13	-49.7	0.30	-32.5
Class E	1.05	-65.2	X	X	2.43	-62.7	0.65	-39	X	X	X	X	1.53	-64	1.03	-65.2	0.87	-53.1
Average	0.95	-54.6	0.73	-20	1.14	-46.9	0.66	-38	1.48	-31	1.78	-20.7	1.14	-59	0.97	-56.2	0.58	-40.5

Tabla 5.22. Comparativa con otros métodos existentes para la configuración intra-only.

Seq.	[46]		[5]		Proposal	
	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT
Class A	0.37	-29.3	0.67	-56.68	0.40	-41.3
Class B	0.72	-30.1	1.17	-50.18	0.82	-42.3
Class C	0.49	-20.5	0.44	-47.91	0.53	-33.5
Class D	0.3	-19.7	0.33	-44.25	0.30	-32.5
Class E	0.67	-41	1.06	-62.60	0.87	-53.1
Average	0.51	-28.12	0.73	-52.32	0.58	-40.5

Tabla 5.23. Comparativa con otros métodos existentes para la configuración low-delay P.

Method	Metric	Class A	Class B	Class C	Class D	Average
[60]	BD-Rate	2.04	2.05	2.15	1.56	1.95
	ΔT	-44.59	-47.04	-36.53	-32.14	-40.07
[59]	BD-Rate	1.56	1.18	2.68	0.77	1.55
	ΔT	-42.73	-40.98	-34.90	-26.65	-36.31
[61]	BD-Rate	2.06	2.94	1.73	1.47	2.05
	ΔT	-58.75	-62.90	-53.60	-52.85	-57.03
[62]	BD-Rate	1.56	1.34	1.24	1.07	1.30
	ΔT	-37.25	-45.68	-34.35	-32.25	-37.38
[63]	BD-Rate	1.50	1.97	1.61	2.08	1.79
	ΔT	-50.70	-55.13	-45.53	-42.40	-48.44
[7]	BD-Rate	X	1	0.8	1.3	1.03
	ΔT	X	-20.9	-18	-17.2	-18.7
[18]	BD-Rate	X	0.7	1	0.8	0.83
	ΔT	X	-15	-14	-13	-14
[46]	BD-Rate	1.23	1.37	1.07	1.30	1.24
	ΔT	-36.8	-46.62	-36.32	-37.18	-39.23
Proposal	BD-Rate	1.10	1.14	0.68	0.58	0.87
	ΔT	-26.79	-32.02	-21.68	-20.60	-25.27

Tabla 5.24. Comparativa con otros métodos existentes para la configuración random-access.

Method	Metric	Class A	Class B	Class C	Class D	Average
[60]	BD-Rate	2.75	2.67	1.62	1.11	2.03
	ΔT	-44.95	-43.38	-38.50	-37.20	-41
[59]	BD-Rate	2.26	4.76	3.91	1.28	3.05
	ΔT	-45.20	-44.90	-38.88	-31.73	-40.17
[61]	BD-Rate	2.86	3.59	2.58	1.77	2.7
	ΔT	-62.95	-65.23	-57.03	-57.18	-60.59
[62]	BD-Rate	1.35	2.09	1.72	1.14	1.57
	ΔT	-45.85	-54.93	-39.00	-35.93	-43.92
[63]	BD-Rate	1.84	3.30	1.86	1.60	2.15
	ΔT	-57.90	-61.58	-50.40	-50.70	-55.14
[7]	BD-Rate	0.4	0.7	0.8	0.9	0.7
	ΔT	-19.9	-19.6	-16.7	-16.7	-18.22
[18]	BD-Rate	1	0.8	1.3	1.2	1.07
	ΔT	-18	-16	-16	-15	-16.25
Proposal	BD-Rate	1.05	1.30	0.98	0.75	1.02
	ΔT	-26.72	-31.30	-22.05	-21.38	-25.36

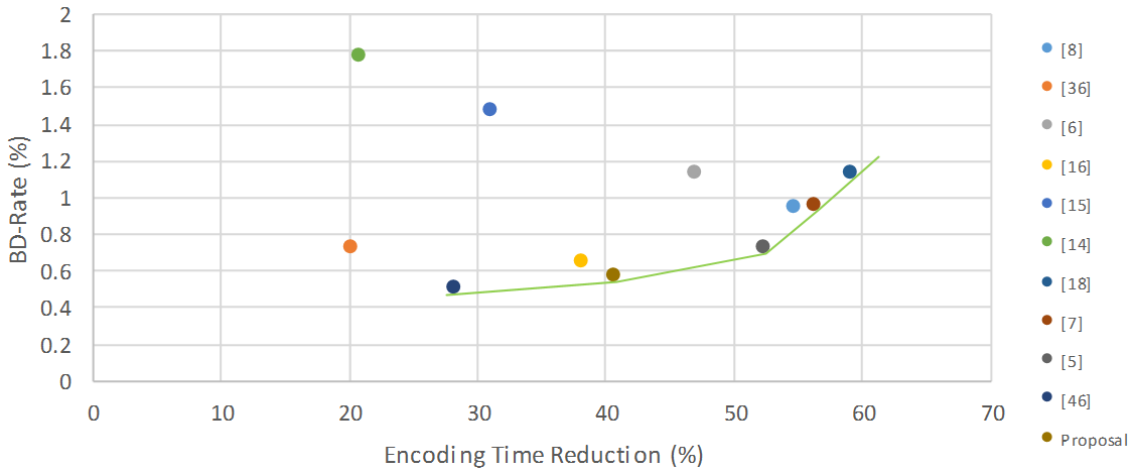


Figura 5.29. Comparativa con otros métodos existentes basada en el frente de Pareto para la configuración intra-only.

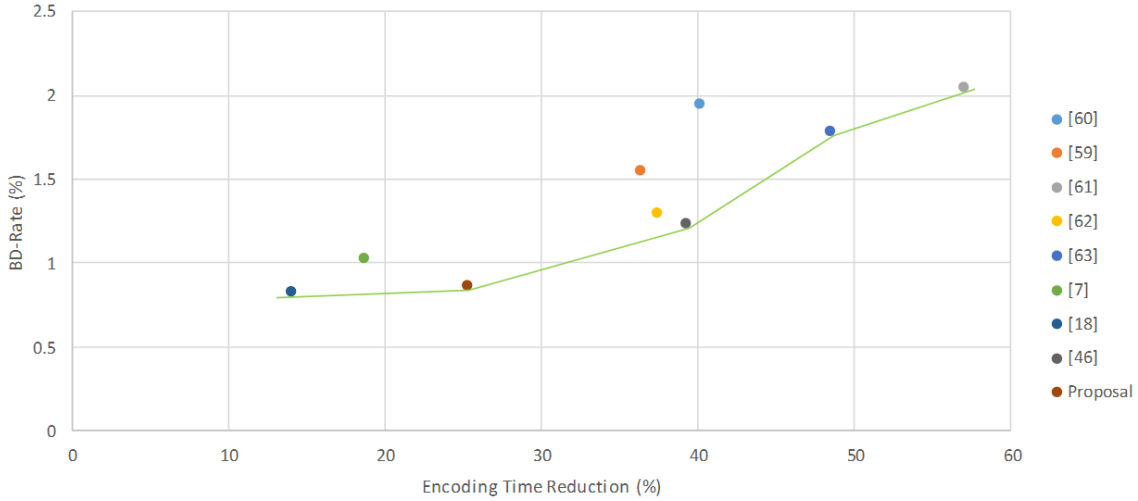


Figura 5.30. Comparativa con otros métodos existentes basada en el frente de Pareto para la configuración low-delay P.

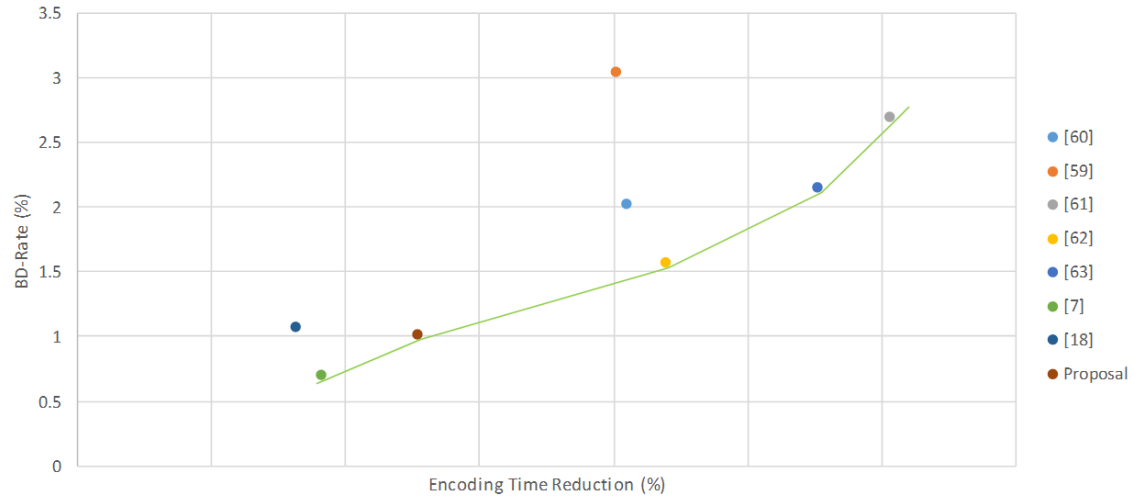


Figura 5.31. Comparativa con otros métodos existentes basada en el frente de Pareto para la configuración random-access.

5.6.1.6 Mejoras subjetivas de la calidad visual

En este apartado se muestran los resultados obtenidos tras incluir mejoras en la calidad subjetiva en el flujo de codificación propuesto. Como se indicó anteriormente, las mejoras subjetivas fueron deshabilitadas en pruebas anteriores debido a que suponen una manipulación de la imagen de entrada o una modificación del valor de cuantificación que se aplica, lo cual llevaría a establecer comparativas equívocas. Los resultados obtenidos haciendo uso de estos algoritmos fueron publicados en [47] y [49].

5.6.1.6.1 Modificación del valor de cuantificación

En la Tabla 5.25 se muestran los resultados en términos de calidad visual obtenidos para diversas secuencias codificadas a diferentes bitrates. En la primera columna se indica el bitrate utilizado, en la siguiente columna se indica el valor MOS alcanzado por la referencia y a continuación, el valor MOS logrado por la propuesta tras aplicar un decremento de QP en regiones que presentan texturas homogéneas. La referencia utilizada es el codificador HM sin modificar el valor de QP que proporciona el RC. En la Tabla 5.25 se han añadido unas columnas indicando el incremento medio y máximo obtenido en cada prueba utilizando la métrica SSIMb. Se ha utilizado la métrica SSIMb al ser una métrica que aplicada sobre secuencias de vídeo codificadas se aproxima con bastante eficacia a la opinión subjetiva de un usuario que visualizase las secuencias, tal y como se demuestra en el trabajo presentado en [39].

Para la realización de estas pruebas, el algoritmo RC es habilitado y todos los planos de cada secuencia son codificados para permitir un ajuste preciso del bitrate, ya que el RC del software HM requiere de cierto tiempo para comenzar a funcionar correctamente y alcanzar la tasa objetivo. Todas las pruebas se ejecutaron haciendo uso de la configuración random-access de manera que trabajen conjuntamente slices tipo I, P y B.

Como se puede observar en la Tabla 5.25, la mejora subjetiva propuesta supera el valor MOS obtenido por la referencia en la mayoría de los casos, logrando como peor resultado un valor MOS igual al de la referencia. En la Tabla 5.25 también se observa que la mejora máxima de calidad utilizando la métrica SSIMb es bastante superior que los valores medios obtenidos, lo cual provoca una percepción de mejora global en la secuencia de vídeo desde el punto de vista del HVS, ya que determinadas partes de la secuencia de vídeo son considerablemente mejoradas.

Tabla 5.25. Resultados obtenidos reduciendo el valor de QP en regiones con texturas homogéneas.

Sequence	Bitrate	Reference MOS Score	Proposal MOS Score	Average ΔSSIMb (%)	Max. ΔSSIMb (%)
blue_sky 1080p25	500 Kbps	3	4	1.5	9.07
	750 Kbps	4	4	1.34	7.01
	1 Mbps	4	5	1.04	6.76
pedestrian_area 1080p25	500 Kbps	2	3	2.48	9.5
	750 Kbps	3	4	2.01	9.04
	1 Mbps	4	4	1.75	7.43
rush_hour 1080p25	500 Kbps	4	4	2.4	5.6
	750 Kbps	4	5	2.34	4.99
	1 Mbps	5	5	1.29	3.78
aspen 1080p30	500 Kbps	1	1	2.22	10.89
	750 Kbps	2	2	2.12	10.5
	1 Mbps	2	3	1.04	10.03

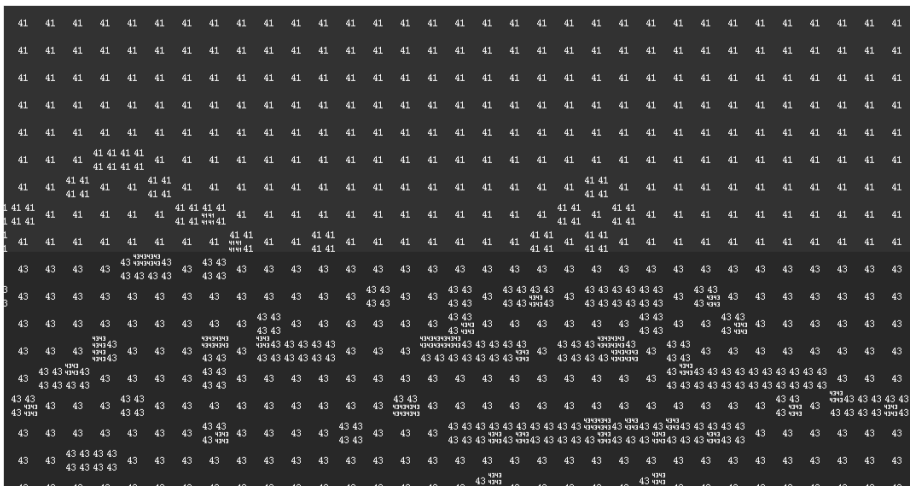
Snow_mountain 1080p30	500 Kbps	2	2	2.37	8.01
	750 Kbps	3	3	2.04	6.27
	1 Mbps	3	4	1.96	5.44
sunflower 1080p25	500 Kbps	4	4	0.98	2.53
	750 Kbps	4	5	0.87	2.22
	1 Mbps	5	5	0.55	1.97
in_to_tree 1080p50	500 Kbps	2	2	2.63	9.89
	750 Kbps	2	3	2.25	7.36
	1 Mbps	3	3	2.01	6.99

La mejora subjetiva propuesta contribuye a la eliminación de efectos indeseables debidos a la codificación tales como blurring, blocking y banding en las regiones que contienen texturas homogéneas. En la Figura 5.32 se observa cómo se reducen los artefactos en el cielo en la secuencia *snow_mountain* tras aplicar la reducción de QP propuesta, lo que mejora la calidad visual subjetiva que se percibe. La mejora obtenida para el plano que se muestra en la Figura 5.32 en términos de SSIMb es cercana al 8%. El valor de MOS obtenido para esta secuencia sin las modificaciones propuestas fue de 3 (los artefactos son levemente molestos). Tras aplicar la reducción de QP en regiones con texturas homogéneas el valor MOS se incrementó a 4 (artefactos perceptibles, pero no molestos).

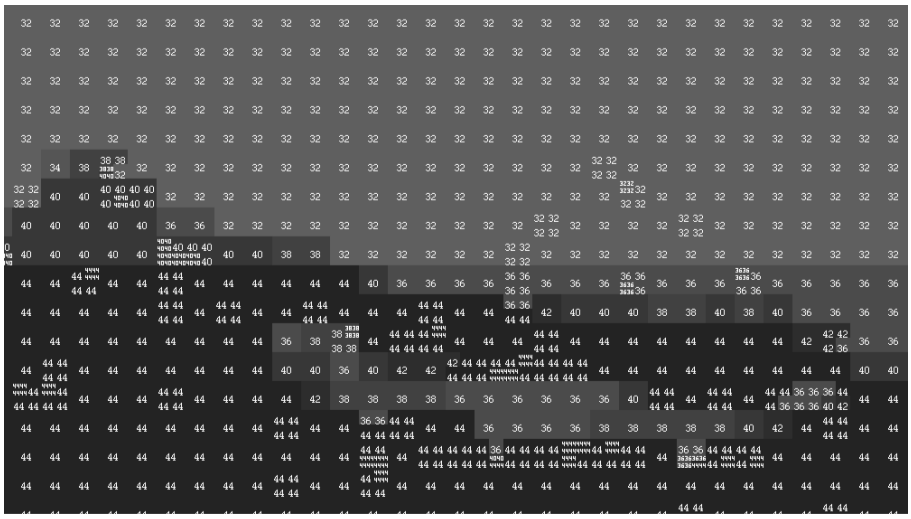
En la Figura 5.32, también se muestra la distribución de valores de QP sin aplicar la mejora propuesta (Figura 5.32.c) y tras aplicar la mejora (Figura 5.32.d). Inicialmente los valores de QP_R que se aplican son 41 y 43. En la imagen que se muestra en la Figura 5.32.d, se puede apreciar cómo tras aplicar la propuesta el valor de QP_S es igual a 32 para el cielo y 36 o 38 para las zonas con sombra de la montaña. Finalmente, se puede observar en esta misma imagen cómo el algoritmo RC ha incrementado el valor de QP_R para el resto de la imagen.

a)

b)



c)



d)

Figura 5.32. Secuencia Snow Mountain codificada utilizada la configuración random-access a 1Mbps.

a) Imagen descodificada sin decremento de QP,
b) Imagen descodificada aplicando decremento de QP en regiones con texturas homogéneas,
c) Distribución de valores de QP sin aplicar decremento,
d) Distribución de valores de QP aplicando decremento de QP en regiones con texturas homogéneas.

5.6.1.6.2 Eliminación de información redundante

El algoritmo de decremento del valor de QP en regiones con texturas homogéneas es complementado por medio de un filtrado aplicado sobre las regiones que presentan una distribución caótica, con vistas a reducir la cantidad de detalle que el codificador necesita procesar. Dichas regiones presentan una gran cantidad de coeficientes de alta frecuencia que no son percibidos por el ojo humano y que pueden ser eliminados sin afectar considerablemente a la calidad subjetiva percibida por el usuario. Reduciendo la información no perceptible de la imagen de entrada que el codificador debe procesar se reduce el número de bits que éste emplea en dichas zonas y permite destinar estos bits a otras regiones de la imagen. Como se comentó anteriormente, la aplicación de decrementos de QP en regiones con texturas homogéneas provoca un leve aumento en el número de bits generado, que puede ser compensado con la reducción en bits conseguida tras filtrar regiones con texturas caóticas, lo cual aumentará aún más la calidad subjetiva global de la secuencia de vídeo.

El algoritmo en cuestión ha sido evaluado implementando el análisis de texturas y el proceso de filtrado en una GPU de la familia Intel HD Graphics 4600, en concreto los resultados que se muestran en este apartado se han obtenido utilizando un microprocesador Intel Core i5-4460@3.4GHz utilizando OpenCL v1.1 [35]. Los tiempos de ejecución asociados a estos procesos son 7.67 ms en media al ser ejecutados sobre planos de resolución 1080p y 3.2 ms para la resolución 720p. El proceso de detección de texturas homogéneas consume 630 µs para planos de resolución 720p y 1.15 ms para la resolución 1080p. La implementación se realizó trabajando únicamente con las imágenes de entrada

e introduciendo un plano de retardo en el pipeline de codificación para no introducir esperas innecesarias en la comunicación GPU-CPU.

En la Tabla 5.26 se muestran los resultados obtenidos para las secuencias HD especificadas en las condiciones de test de [33] utilizando bajos bitrates, puesto que a tasas de transmisión bajas es cuando más se aprecian los artefactos debidos a la compresión. En la Tabla 5.26 se puede apreciar cómo el ahorro de bits debido al filtrado realizado permite alcanzar una mayor calidad visual al mismo tiempo que se mantiene el bitrate objetivo respecto al caso base en el que el codificador no incluye ninguna mejora de calidad subjetiva. En dicha tabla se observan máximos de hasta un 12% de mejora visual. También se observa cómo la mejora disminuye según aumenta el bitrate, ya que cuanto mejor se ve la imagen, menor margen de mejora existe.

En la Figura 5.33 se puede apreciar cómo el algoritmo propuesto reduce los artefactos debidos a blurring, blocking, efecto mosquito y banding para el plano 240 de la secuencia *in_to_tree* codificada a 512 Kbps. Para este plano en concreto, la mejora obtenida cuantificada con la métrica SSIMb es cercana al 11.5%.

El algoritmo de filtrado propuesto utiliza la clasificación de texturas homogéneas y el cálculo de la amplitud del gradiente de Sobel. Estos cálculos son utilizados por algoritmos de optimización de la decisión del tamaño de bloque a usar, como el algoritmo diseñado para tal propósito en esta Tesis basado en texturas homogéneas o el algoritmo basado en gradientes presentado en [10]. Por lo tanto, además de una reducción en la carga computacional de la codificación se puede conseguir una mejora de calidad visual subjetiva. La mejora en la carga computacional obtenida tras integrar ambos algoritmos, utilizando las secuencias HD bajo estudio, se muestra en la Tabla 5.27.

Tabla 5.26. Resultados obtenidos al complementar el algoritmo de reducción de QP en regiones con texturas homogéneas con un filtrado sobre las regiones con texturas caóticas.

Sequences	Bitrate	Average ΔSSIMb (%)	Maximum ΔSSIMb (%)	MOS Ref	MOS Proposal
Class B (1080p)	512 Kbps	2.34	12.03	2	2.72
	750 Kbps	2.12	10.07	2.44	3.18
	1 Mbps	2.01	9.11	3	3.56
	2 Mbps	1.5	9.01	3.4	4.1
	3 Mbps	1.06	6.01	3.98	4.46
Class E (720p)	512 Kbps	2.21	11.22	3	3.3
	750 Kbps	1.94	8.71	3	4
	1 Mbps	1.34	7.14	4	4.35
	2 Mbps	1.23	7.27	4	4.23
	3 Mbps	0.076	4.34	4.65	4.8

Tabla 5.27. Integración de la mejora perceptual propuesta con algoritmos de optimización de la decisión de tamaño de bloque.

Sequence	BD_BR (%)	ΔEncoding Time (%)
Class B (1080p)	0.8	-25,38
Class E (720p)	0.8	-36,67

a)

b)

Figura 5.33. Comparación visual para el plano 240 de la secuencia in_to_tree (1080p).
a) Imagen descodificada sin mejora perceptual,
b) Imagen descodificada tras aplicar la mejora propuesta.

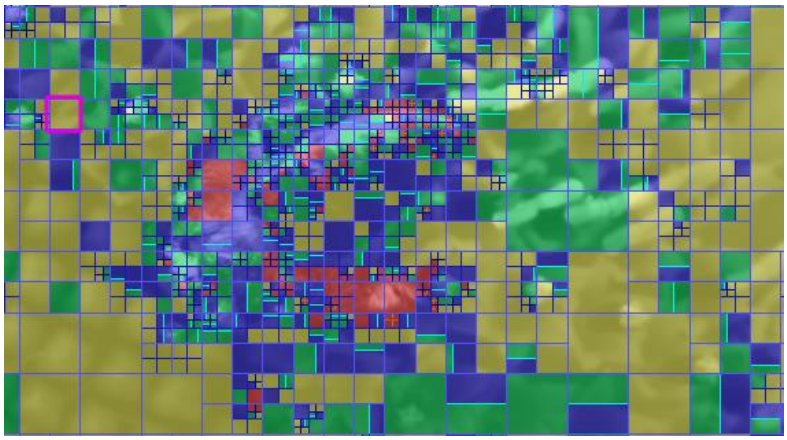
5.6.2 Optimizaciones basadas en la homogeneidad temporal

En la Tabla 5.28 se muestran los resultados publicados en [50] que fueron obtenidos aplicando únicamente el algoritmo de decisión de tamaño de bloque basado en la detección de regiones de la imagen clasificadas como temporalmente homogéneas.

La estimación de movimiento necesaria para la obtención del mapa de vectores de movimiento que se requiere para el cálculo de la desviación absoluta media requiere 8.1 ms para imágenes de resolución 720p y 17.4 ms para imágenes 1080p. Estas medidas de tiempo han sido obtenidas en el modelo de GPU HD Graphics 530 de Intel. Repitiendo las

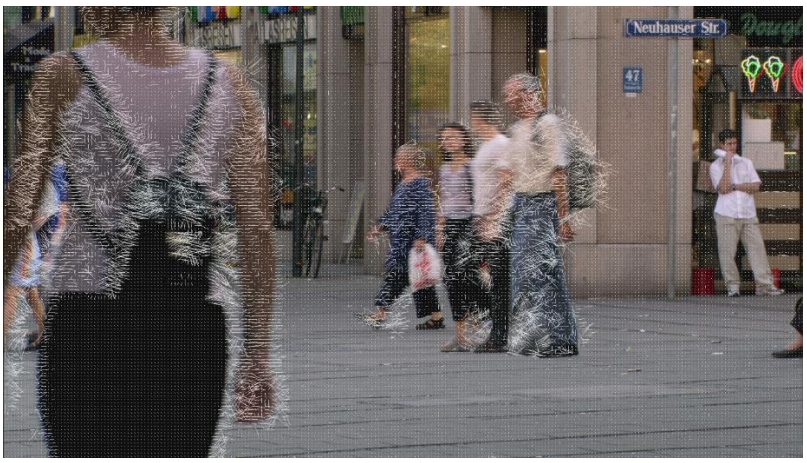
mismas pruebas en el modelo HD Graphics 4600, se obtienen 8.2 y 18.2 ms, respectivamente. Por lo tanto, es posible realizar hasta dos estimaciones de movimiento en el caso del formato de vídeo 1080p25 donde se dispone de hasta 40 ms por plano para alcanzar ejecución en tiempo real, y únicamente una para el caso de 1080p50 donde se dispone de 20 ms por plano. En el algoritmo propuesto, se realiza una única estimación de movimiento aplicada sobre el plano anterior de la secuencia de vídeo (en orden de presentación) pero realizada de una manera exhaustiva evaluando todos los puntos existentes en la ventana de búsqueda. En la Figura 5.34.a se muestra el mapa de vectores de movimiento obtenido para el plano 31 de la secuencia *sunflower* tras ampliar la imagen. En la Figura 5.34.b, se muestra el resultado obtenido por el algoritmo de decisión de tamaño de bloque basado en la clasificación de regiones temporalmente homogéneas. Como se puede apreciar, en los bloques que presentan homogeneidad temporal se han escogido tamaños grandes de bloque mientras que los tamaños de bloque pequeño son escogidos para regiones de la imagen que no presentan movimiento uniforme como es el caso de la abeja. En la Figura 5.35 se muestra un análisis equivalente para el plano 51 de la secuencia *pedestrian*. En la Tabla 5.28 se muestran los resultados en términos de incremento de bitstream, incremento de calidad visual empleando la métrica PSNR y el incremento en tiempos de codificación. Se muestra la media de las pruebas realizadas implementando el algoritmo propuesto en el software HM v16.2 y utilizando nueve valores de QP diferentes: 10, 15, 20, 25, 30, 35, 40, 45 y 50. Como se puede observar, se han reducido los tiempos de ejecución hasta en un 38% en el mejor caso. Existen secuencias en las que se reduce el tamaño del bitstream, siendo 0.224% la mayor reducción. El mayor incremento en el tamaño del bitstream es de 0.93%, mientras que la pérdida de calidad es imperceptible, siendo 0.045 dB la mayor pérdida. Los tiempos de ejecución indicados han sido obtenidos sobre el microprocesador Intel Core i5-4460@3.4GHz utilizando un único core, puesto que el software HM no soporta codificación multihilo.

a)

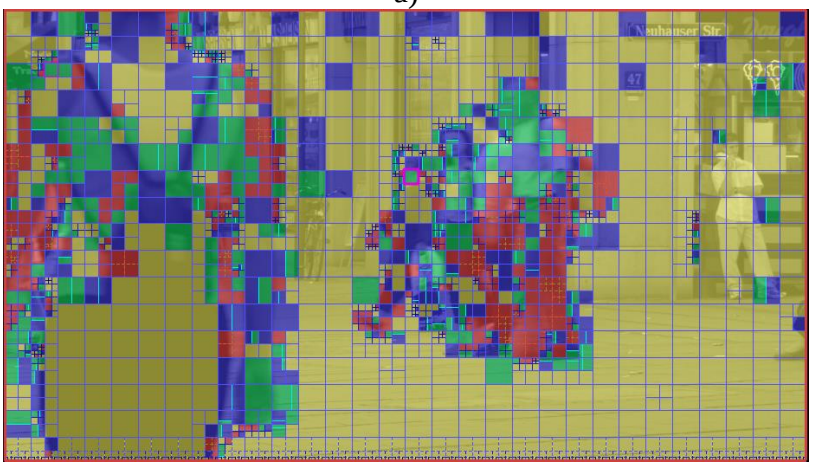


b)

Figura 5.34. Resultados tras aplicar la detección de homogeneidad temporal (secuencia sunflower).
a) Mapa de vectores obtenido para el plano 31,
b) Resultado del algoritmo de decisión de tamaño de bloque propuesto.



a)



b)

Figura 5.35. Resultados tras aplicar la detección de homogeneidad temporal (secuencia pedestrian).
a) **Mapa de vectores obtenido para el plano 51 de la secuencia pedestrian,**
b) Resultado del algoritmo de decisión de tamaño de bloque propuesto.

Tabla 5.28. Resultados obtenidos por el algoritmo de decisión de tamaño de bloque basado en la detección de homogeneidad temporal para la configuración low-delay P.

Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔEncoding Time (%)
MobileCalendar 720p50	0.109	-0.034	-15.325
ParkRun 720p50	0.029	-0.021	-14.152
Shields 720p59	0.118	-0.012	-6.736
Aspen 1080p30	0.0513	-0.0002	-25.118
BlueSky 1080p25	0.036	-0.005	-30.991
CrowdRun1080p50	0.93	-0.020	-7.413
DucksTakeOff 1080p50	0.033	-0.006	-3.61
InToTree 1080p50	-0.052	-0.006	-26.573
ParkJoy 1080p50	0.456	-0.015	-4.591
PedestrianArea 1080p25	0.538	-0.012	-23.49
RedKayak 1080p25	0.008	0.096	-21.97
Riverbed 1080p25	-0.068	0.006	-2.382
RushHour 1080p25	0.028	-0.001	-21.416
SnowMountain 1080p25	0.777	-0.045	-38.267
Stockholm 720p60	0.418	-0.016	-12.113
Sunflower 1080p25	0.108	-0.005	-17.392
Tractor 1080p25	-0.224	-0.0008	-5.272

Al tratarse de una optimización basada en la homogeneidad temporal sólo puede aplicarse sobre slices tipo P o B, donde existen CU tipo inter en las que en la codificación se tiene en cuenta la información relativa al movimiento. Sin embargo, esta optimización es totalmente compatible con la previamente comentada basada en la detección de texturas homogéneas aplicada tanto a CU tipo intra como inter. Por ello, en las siguientes tablas y figuras que se presentan se ha decidido integrar ambos análisis de la imagen en busca de detectar tanto regiones temporalmente homogéneas como aquellas que presenten texturas homogéneas. Si una CU es perteneciente a alguna de estas regiones entonces, el algoritmo de división recursiva de la imagen en bloques de tamaño inferior es detenido. En la Figura 5.36, se muestra el flujo de codificación que a continuación se procederá a evaluar y cuyos resultados se publicaron en [46].

La Tabla 5.29 muestra el resultado obtenido aplicando únicamente la optimización en CU que presentan texturas homogéneas para las mismas condiciones de test utilizadas para la Tabla 5.28. En la Tabla 5.30 se muestran los resultados para la configuración low-delay P tras aplicar el algoritmo de decisión de tamaño de bloque basado tanto en la detección de texturas homogéneas como en la detección de homogeneidad temporal. Como se puede observar en la Tabla 5.29, explotando únicamente la homogeneidad espacial basada en la detección de texturas homogéneas, la reducción en el tiempo de codificación oscila entre 2.1% y 39.3% dependiendo del número de CU clasificadas de este modo, con una pérdida de calidad imperceptible y un aumento en el tamaño del bistream despreciable. De hecho, para muchas secuencias el valor de la media mostrado es negativo debido a que para varios valores de QP el tamaño del bitstream es reducido. En el mejor caso el tamaño del bitstream es reducido hasta un 0.72%, siendo 0.25% el peor incremento existente.

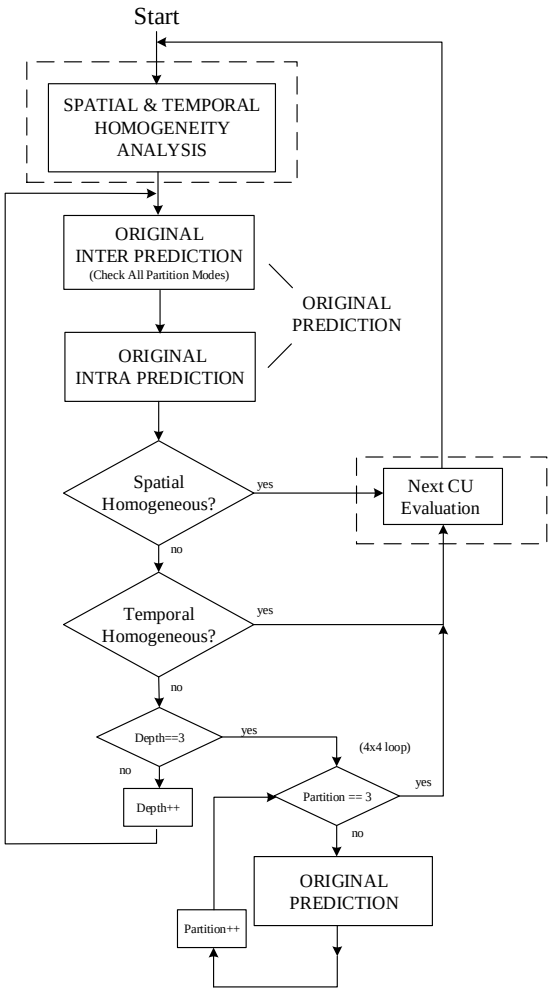


Figura 5.36. Flujo de codificación utilizando el algoritmo de decisión de tamaño de bloque optimizado basado en la clasificación de texturas homogéneas y regiones temporalmente homogéneas.

En la Tabla 5.30 se observa cómo la inclusión de la explotación de la homogeneidad temporal conlleva en determinadas secuencias una importante reducción en los tiempos de codificación, alcanzando el valor de 42.35% en el mejor de los casos. El tamaño del bitstream sigue siendo reducido en general, siendo 0.8% el mayor incremento y 0.53% el mayor decremento conseguido. La mayor pérdida de calidad es 0.11 dB, lo cual es prácticamente imperceptible. En general, la explotación de la homogeneidad temporal mejora los resultados en aquellas secuencias en las cuales no existen muchas texturas homogéneas, como las secuencias *MobileCalendar* o *Parkrun*. En otras secuencias como *SnowMountain*, es posible alcanzar una importante mejora adicional pasando de una reducción de 26.87% hasta lograr un 42.35%, lo que supone un incremento en la reducción del tiempo de codificación de 15.48%.

En la Figura 5.37 se muestra una detallada comparativa entre el software HM v16.2 sin ninguna modificación respecto al mismo software incluyendo el algoritmo de decisión de tamaño de bloque optimizado, en términos de PSNR, número de bits generados y tiempo de ejecución para cada plano de la secuencia *SnowMountain* codificada empleando un valor de QP=25.

En la Tabla 5.31 se muestra una comparativa entre los trabajos más relevantes del estado del arte con la propuesta realizada para la configuración low-delay P utilizando las condiciones de test recomendadas por JCT-VT [33].

Tabla 5.29. Resultados obtenidos por el algoritmo de decisión de tamaño de bloque basado en la detección de texturas homogéneas para la configuración low-delay P.

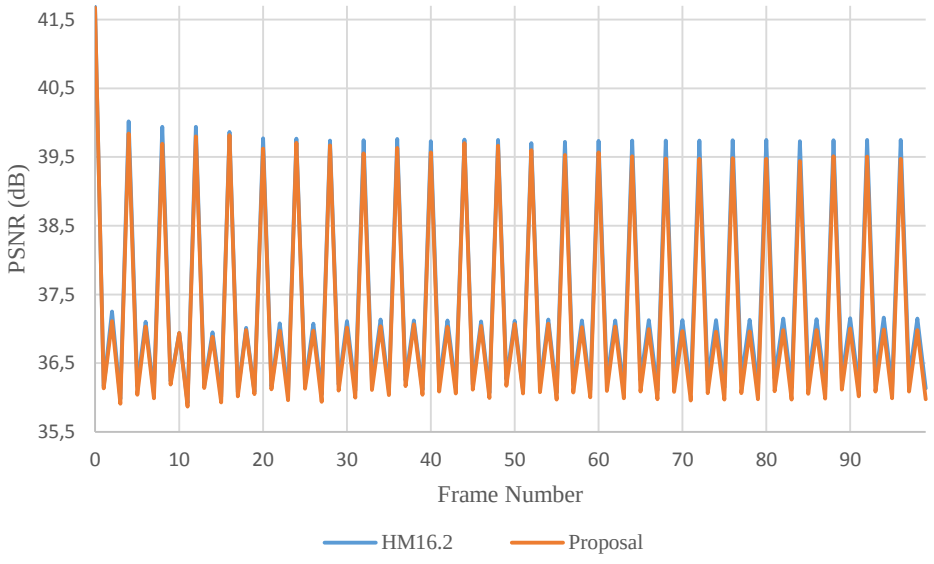
Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔTime (%)
MobileCalendar 720p50	0.254	-0.005	-3.343
ParkRun 720p50	-0.088	-0.008	-2.138
Shields 720p59	-0.073	-0.008	-2.937
Aspen 1080p30	-0.427	-0.030	-39.318
BlueSky 1080p25	-0.183	-0.017	-31.735
CrowdRun1080p50	-0.079	-0.014	-9.076
DucksTakeOff 1080p50	-0.020	-0.004	-3.278
InToTree 1080p50	-0.719	-0.025	-28.148
ParkJoy 1080p50	-0.047	-0.001	-2.261
PedestrianArea 1080p25	0.100	-0.008	-20.502
RedKayak 1080p25	-0.088	-0.035	-25.892
Riverbed 1080p25	-0.086	0.004	-5.019
RushHour 1080p25	-0.095	-0.012	-22.768
SnowMountain 1080p25	0.014	-0.055	-26.873
Stockholm 720p60	0.157	-0.011	-14.289
Sunflower 1080p25	-0.380	-0.004	-15.431
Tractor 1080p25	-0.082	-0.065	-6.172

Tabla 5.30. Resultados obtenidos por el algoritmo de decisión de tamaño de bloque basado en la tanto en la detección de texturas homogéneas como de regiones temporalmente homogéneas para la configuración low-delay P.

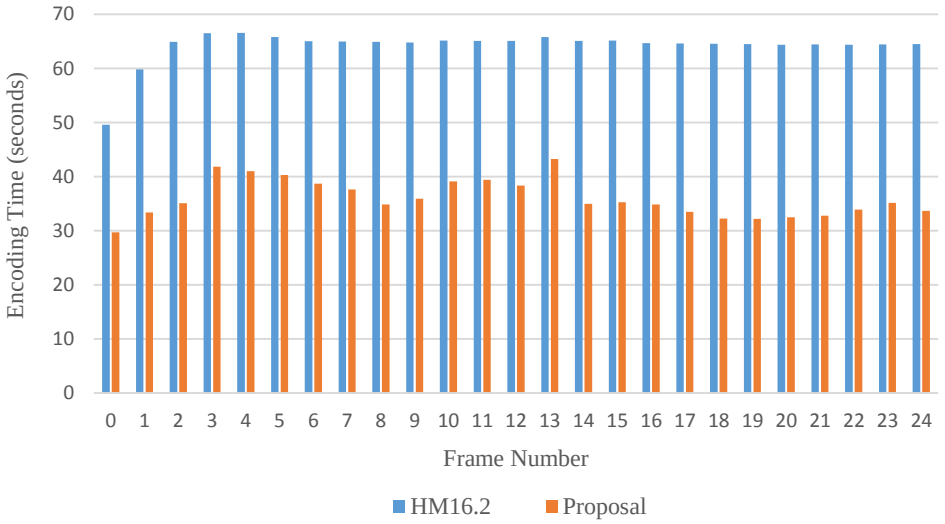
Sequence	ΔBitstream (%)	ΔPSNR (dB)	ΔTime (%)
MobileCalendar 720p50	0.233	-0.021	-18.770
ParkRun 720p50	0.338	-0.044	-16.410
Shields 720p59	0.009	-0.013	-7.533
Aspen 1080p30	-0.431	-0.030	-39.906
BlueSky 1080p25	-0.183	-0.015	-32.127
CrowdRun1080p50	-0.009	-0.023	-13.913
DucksTakeOff 1080p50	-0.018	-0.003	-5.110
InToTree 1080p50	-0.530	-0.028	-31.829
ParkJoy 1080p50	-0.046	-0.004	-5.759
PedestrianArea 1080p25	0.142	-0.008	-25.044
RedKayak 1080p25	-0.105	-0.036	-27.852
Riverbed 1080p25	-0.087	0.004	-5.503
RushHour 1080p25	-0.084	-0.012	-24.165
SnowMountain 1080p25	0.806	-0.114	-42.351
Stockholm 720p60	0.075	-0.011	-15.111
Sunflower 1080p25	-0.250	-0.004	-21.873
Tractor 1080p25	-0.177	0.003	-10.430

Tabla 5.31. Comparativa con otros métodos existentes para la configuración low-delay P.

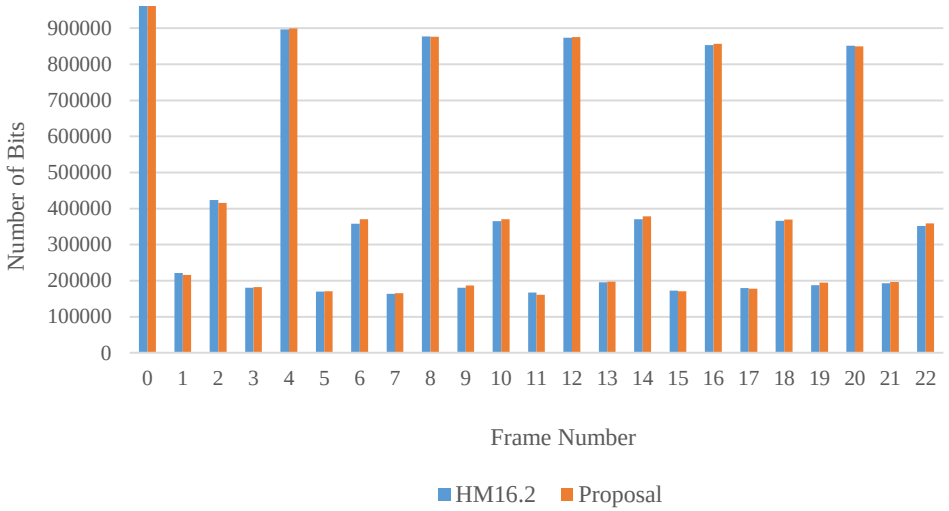
Method	Metric	Class A	Class B	Class C	Class D	Average
[60]	BD-Rate	2.04	2.05	2.15	1.56	1.95
	ΔT	-44.59	-47.04	-36.53	-32.14	-40.07
[59]	BD-Rate	1.56	1.18	2.68	0.77	1.55
	ΔT	-42.73	-40.98	-34.90	-26.65	-36.31
[61]	BD-Rate	2.06	2.94	1.73	1.47	2.05
	ΔT	-58.75	-62.90	-53.60	-52.85	-57.03
[62]	BD-Rate	1.56	1.34	1.24	1.07	1.30
	ΔT	-37.25	-45.68	-34.35	-32.25	-37.38
[63]	BD-Rate	1.50	1.97	1.61	2.08	1.79
	ΔT	-50.70	-55.13	-45.53	-42.40	-48.44
[7]	BD-Rate	X	1	0.8	1.3	1.03
	ΔT	X	-20.9	-18	-17.2	-18.7
[18]	BD-Rate	X	0.7	1	0.8	0.83
	ΔT	X	-15	-14	-13	-14
Proposal	BD-Rate	1.23	1.37	1.07	1.30	1.24
	ΔT	-36.8	-46.62	-36.32	-37.18	-39.23



a)



b)



c)

Figura 5.37. Comparativa HM v16.2 vs Propuesta.
Secuencia SnowMountain empleando QP=25. En términos de:
a) PSNR. b) Tiempo de ejecución c) Bits generados.

En la Figura 5.38 se demuestra que la propuesta pertenece al frente de Pareto en el dominio BD-Rate/Encoding Time Reduction por lo que se puede considerar como una solución óptima al problema abordado. Como muestra el gráfico, la solución de compromiso adoptada proporciona la mayor calidad para reducciones en tiempo de ejecución superior al 35%.

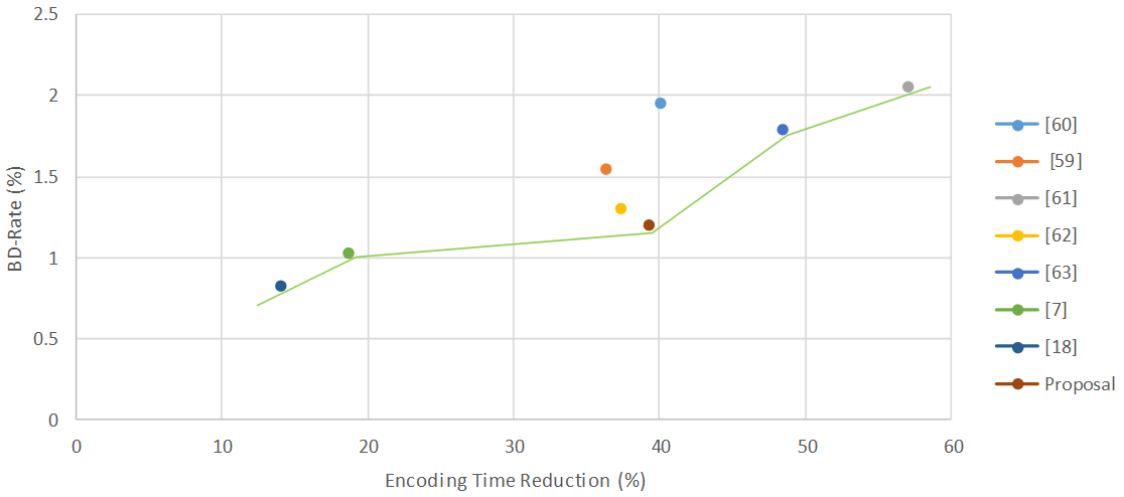


Figura 5.38. Comparativa con otros métodos existentes basada en el frente de Pareto para la configuración low-delay P.

5.6.3 Verificación del flujo de codificación completo

A continuación, se procederá a evaluar el flujo de codificación completo mostrado en las Figuras 5.23 y 5.24. Por lo tanto, se evaluará de manera conjunta el funcionamiento de tanto las optimizaciones aplicadas en regiones clasificadas como espacialmente homogéneas como aquellas aplicadas a regiones clasificadas como temporalmente homogéneas. Las zonas de la imagen que han sido clasificadas como espacialmente homogéneas a su vez pueden pertenecer a dos categorías diferenciadas, regiones que presentan texturas homogéneas y aquellas que presentan una direccionalidad espacial claramente definida. Por lo tanto, el flujo de codificación completo incluye las siguientes aportaciones:

1. Detección de regiones que presentan texturas homogéneas mediante un análisis realizado sobre la imagen de entrada utilizando las unidades lógicas de la GPU.
2. Detección de regiones temporalmente homogéneas mediante el análisis de un mapa de vectores de movimiento obtenido tras aplicar una estimación de movimiento sobre la imagen de entrada. La estimación de movimiento utiliza módulos hardware específicos basados en H.264 de las GPU de Intel. El análisis del mapa de vectores de movimiento y posterior clasificación, se realiza empleando las unidades lógicas de la GPU.
3. Aplicación de un algoritmo de decisión de tamaño de bloque optimizado tanto en CU tipo intra como inter en regiones que presentan texturas homogéneas.
4. Aplicación de un algoritmo de decisión de tamaño de bloque y de decisión de modo de partición inter en CU tipo inter en regiones temporalmente homogéneas.
5. Un algoritmo optimizado de decisión de modo de predicción intra en regiones que presentan texturas homogéneas.
6. Detección de regiones espacialmente homogéneas en las cuales existe una clara direccionalidad espacial predominante haciendo uso de un módulo hardware específico presente en las GPU de Intel para obtener los mejores modos de predicción y tamaños de bloque bajo el estándar H.264. El análisis de los datos, la

adaptación al estándar HEVC y la clasificación en sí, es realizada mediante las unidades lógicas de la GPU.
7. Aplicación de un algoritmo de decisión de tamaño de bloque optimizado en CU tipo intra en regiones que presentan una direccionalidad espacial predominante.
8. Un algoritmo optimizado de decisión de modo de predicción intra en regiones que presentan una direccionalidad espacial predominante, utilizando información derivada de la predicción intra H.264.
9. Un algoritmo de configuración adaptativa de la ventana de búsqueda para la estimación de movimiento en regiones temporalmente homogéneas.
10. Parada prematura aplicada sobre la predicción inter, basada en la comparación de los costes obtenidos para la predicción intra e inter por los módulos hardware H.264 de Intel.
11. Mejoras subjetivas de la calidad visual por medio de una reducción del valor de cuantificación en regiones con texturas homogéneas y un filtrado aplicado sobre regiones con estructura caótica.
12. Algoritmo de reducción de artefactos intrínsecos al HEVC.
13. Evaluación de la calidad basada en HVS.
14. Implementación en un sistema de codificación en tiempo real.

En el primer conjunto de gráficas y tablas que muestran un resumen de los resultados obtenidos se han utilizado las condiciones de test recomendadas por JCT-VT [33] aplicadas sobre el software HM v16.2, con el propósito de establecer comparativas con otros trabajos existentes en la literatura. En la Tabla 5.32 se muestra una primera comparativa para la configuración intra-only y en la Tabla 5.33, una segunda comparativa para la configuración low-delay P. En los resultados que se muestran en las Tablas 5.32 y 5.33, las mejoras subjetivas de la calidad han sido deshabilitadas para poder realizar una comparativa justa, ya que como se comentó previamente, las métricas PSNR y BD-Rate que se emplean para caracterizar a los algoritmos son métricas objetivas, y pueden llevar a una mala interpretación de los resultados obtenidos cuando se aplican para caracterizar mejoras subjetivas de calidad.

En las Figuras 5.39 y 5.40, se muestra que la propuesta pertenece al frente de Pareto en el dominio BD-Rate/Encoding Time Reduction, por lo que dentro de los trabajos que se han usado para la comparativa puede considerarse como una de las soluciones óptimas. Además, la propuesta consigue una de las soluciones de compromiso más equilibradas, ya que obtiene elevadas reducciones en el tiempo de ejecución a la vez que el BD-Rate obtenido es relativamente bajo. En las Figuras 5.39 y 5.40, aparecen los resultados referentes a las primeras publicaciones que se realizaron [46, 47] y cuyos algoritmos fueron complementados con nuevas optimizaciones para mejorar las prestaciones del codificador. Los resultados finales que aparecen en dichas figuras y en las Tablas 5.32 y 5.33, fueron publicados en [45].

Tabla 5.32. Comparativa con otros métodos existentes para la configuración intra-only.

Seq.	[8]		[36]		[6]		[16]		[15] α=0.7		[14]		[18]		[7]		Proposal	
	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT	BD-Rate	ΔT
Class A	0.85	-54.8	0.55	-20.4	1.14	-50	0.39	-37.8	1.41	-35	2.28	-21.9	1.15	-59	0.80	-56.2	0.53	-53.71
Class B	0.69	-58.7	0.70	-20.5	1.26	-53.2	0.56	-40.6	0.79	-24.3	2.16	-28.6	0.94	-61	0.72	-59.4	1.05	-52.82
Class C	0.80	-47.2	0.74	-19.5	0.60	-34.2	0.84	-37.3	1.85	-33.6	1.50	-17.8	1.08	-56.5	1.18	-50.7	0.88	-46.39
Class D	1.37	-46.9	0.94	-19.5	0.25	-34.5	0.84	-35.5	1.86	-31.1	1.19	-14.5	1.00	-54.7	1.13	-49.7	0.55	-44.13
Class E	1.05	-65.2	X	X	2.43	-62.7	0.65	-39	X	X	X	X	1.53	-64	1.03	-65.2	1.16	-61.74
Average	0.95	-54.6	0.73	-20	1.14	-46.9	0.66	-38	1.48	-31	1.78	-20.7	1.14	-59	0.97	-56.2	0.83	-51.76

Tabla 5.33. Comparativa con otros métodos existentes para la configuración low-delay P.

Method	Metric	Class A	Class B	Class C	Class D	Average
[60]	BD-Rate	2.04	2.05	2.15	1.56	1.95
	ΔT	-44.59	-47.04	-36.53	-32.14	-40.07
[59]	BD-Rate	1.56	1.18	2.68	0.77	1.55
	ΔT	-42.73	-40.98	-34.90	-26.65	-36.31
[61]	BD-Rate	2.06	2.94	1.73	1.47	2.05
	ΔT	-58.75	-62.90	-53.60	-52.85	-57.03
[62]	BD-Rate	1.56	1.34	1.24	1.07	1.30
	ΔT	-37.25	-45.68	-34.35	-32.25	-37.38
[63]	BD-Rate	1.50	1.97	1.61	2.08	1.79
	ΔT	-50.70	-55.13	-45.53	-42.40	-48.44
[7]	BD-Rate	X	1	0.8	1.3	1.03
	ΔT	X	-20.9	-18	-17.2	-18.7
[18]	BD-Rate	X	0.7	1	0.8	0.83
	ΔT	X	-15	-14	-13	-14
Proposal	BD-Rate	1.3	1.41	1.09	1.34	1.28
	ΔT	-40.3	-47.52	-35.1	-37.6	-45.13

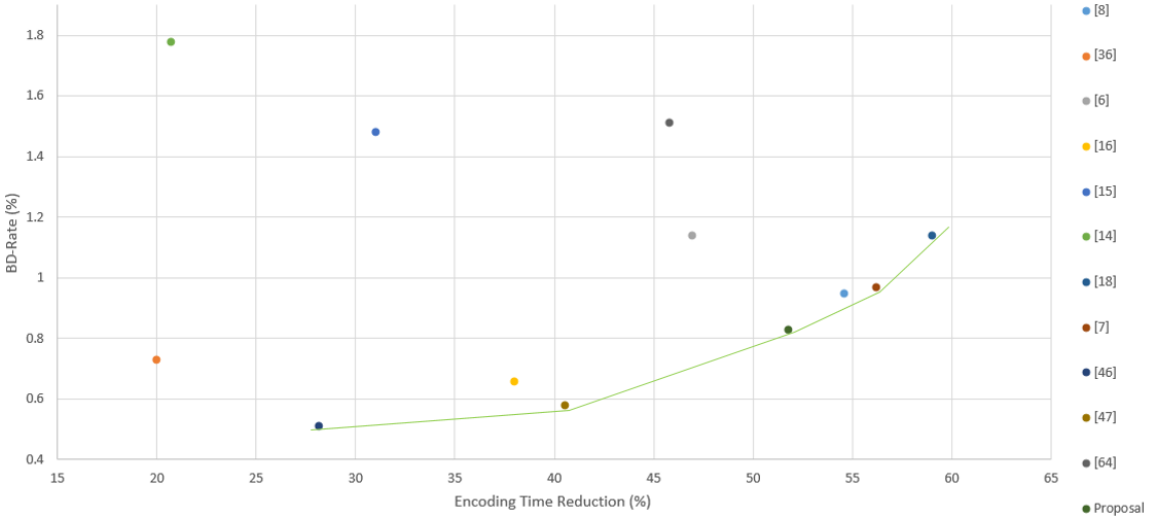


Figura 5.39. Comparativa con otros métodos existentes basada en el frente de Pareto para la configuración intra-only.

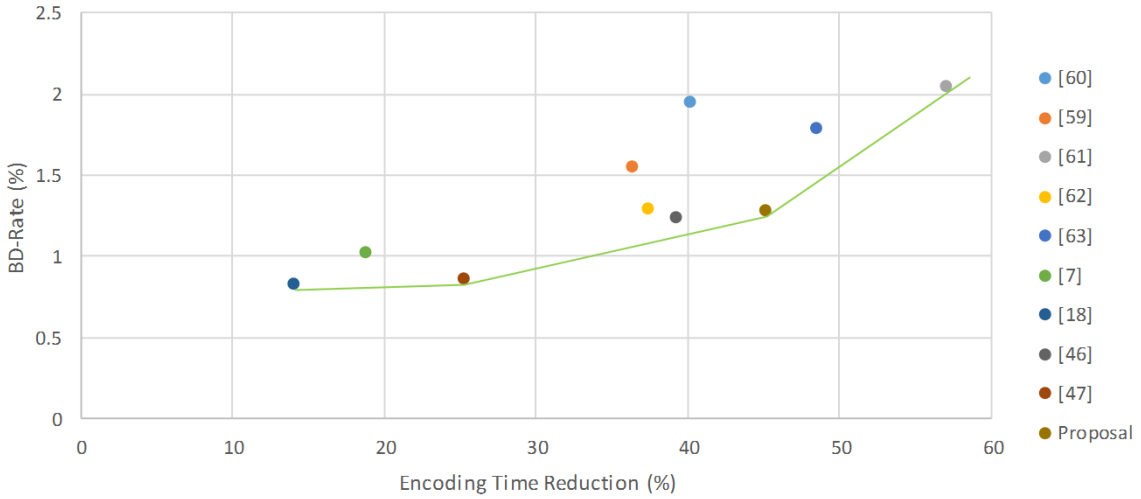


Figura 5.40. Comparativa con otros métodos existentes basada en el frente de Pareto para la configuración low-delay P.

Uno de los principales objetivos que se persiguen en la presente Tesis Doctoral es facilitar la implementación de un codificador HEVC en tiempo real. En términos académicos las publicaciones utilizan para la obtención de los resultados en su mayoría el software de referencia HM, pero como ya se comentó con anterioridad, su ejecución dista enormemente de permitir la codificación en tiempo real. Por el contrario, el software x265 ha sido implementado con vistas a ser utilizado en entornos basados en microprocesador que requieran codificación en tiempo real. Aun así, la codificación es computacionalmente muy demandante por lo que los algoritmos de optimización planteados en apartados anteriores facilitarán enormemente el uso de dispositivos que abaraten el coste final del sistema por medio del uso de microprocesadores/GPU menos potentes. De manera indirecta, al requerirse una potencia de cómputo inferior se reduce el consumo del sistema de codificación empleado.

En la Tabla 5.34 se muestran los resultados obtenidos para distintas secuencias que presentan diversas características integrando los algoritmos propuestos en x265. El caso base respecto al cual se establecen los incrementos porcentuales de las distintas métricas es el software x265 original sin ninguna de las modificaciones planteadas. Los valores que aparecen son la media de los resultados obtenidos empleando diferentes bitrates (5 Mbps, 10 Mbps, 15 Mbps y 20 Mbps). En cada prueba se verificó que el bitrate objetivo era alcanzado y mantenido. Para la caracterización de la calidad visual alcanzada, se han utilizado diversas métricas que contemplan consideraciones perceptuales. Hay que recordar, aunque ya se mencionó con anterioridad, que valores negativos suponen un decremento de la métrica correspondiente, por lo que implicaría una pérdida de calidad a excepción de cuando se utiliza la métrica VQM. Empleando VQM un valor positivo implica una pérdida de calidad.

Como se puede apreciar en la Tabla 5.34 la reducción en tiempos de ejecución oscila entre el 12% y el 42.9% (siendo 23.8% el valor de la media), mientras que la pérdida de calidad es imperceptible. Para completar el estudio se llevó a cabo una comparación subjetiva utilizando MOS, obteniendo como resultado que ninguno de los expertos que visualizaron la comparativa notó diferencia alguna entre los vídeos.

Tabla 5.34. Resultados tras la integración de los métodos propuestos en x265.

Sequence	ΔTime (%)	Δ3SSIM (%)	ΔST-SSIM (%)	ΔMSSIM (%)	ΔVQM (%)	ΔPSNR (%)	ΔPSNR HVSM (%)	ΔVIFP (%)	ΔSSIM (%)
Aspen 1080p30	-27,74	-0.11	-0.1	-0.14	0.12	-0.17	-0.26	-0.84	-0.09
BlueSky 1080p25	-27,51	-0.07	-0.21	-0.2	0.92	-0.37	-0.39	-1.3	-0.04
CrowdRun 1080p50	-19,22	-0.23	-0.64	-0.08	0.51	-0.21	-0.26	-1.98	-0.18
DucksTakeOff 1080p50	-12,12	-0.08	-0.31	-0.03	0.34	-0.24	-0.26	-0.04	-0.06
InToTree 1080p50	-32,18	-0.31	-1.86	-0.12	1.5	-0.61	-0.79	-1.84	-0.30
ParkJoy 1080p50	-34,11	-0.18	-1.32	-0.11	0.99	-0.39	-0.47	-0.71	-0.16
PedestrianArea 1080p25	-19,93	-0.1	-1.19	-0.1	2.78	-0.49	-0.51	-0.97	-0.09
Riverbed 1080p25	-19,46	-0.01	-0.11	0.00	0.76	-0.22	-0.24	-0.03	-0.01
RushHour 1080p25	-16,33	-0.21	-1.01	-0.06	0.12	-0.39	-0.40	-0.04	-0.2
SnowMountain 1080p25	-42,92	-0.17	-0.77	-0.09	1.23	-0.21	-0.13	-0.71	-0.17
Sunflower 1080p25	-17,84	-0.12	-1.1	-0.07	3.16	-0.11	-0.12	-0.53	-0.13
Tractor 1080p25	-15,99	-0.09	-0.86	0.00	0.51	-0.07	-0.12	-0.22	-0.07
Average	-23,78	-0,14	-0,79	-0,08	1,08	-0,29	-0,33	-0,77	-0,13

5.7 Referencias

[1] He G., Zhou D., Goto S. (2013). Transform-based fast mode and depth decision algorithm for HEVC intra prediction. Paper presented at IEEE 10th International Conference on ASIC (ASICON), Shenzhen, China.

[2] Zhang Y., Li Z., Li B. (2012). Gradient-based fast decision for intra prediction in HEVC. Paper presented at Visual Communications and Image Processing (VCIP), San Diego, CA, USA.

[3] Ting Y-C., Chang T-S. (2014). Gradient-based PU size selection for HEVC intra prediction. Paper presented at IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne VIC, Australia.

[4] Tian G., Goto S. (2012). Content adaptive prediction unit size decision algorithm for HEVC intra coding. Paper presented at Picture Coding Symposium (PCS), Krakow, Poland.

[5] Min B., Cheung R. (2014). A fast cu size decision algorithm for HEVC intra encoder. IEEE Transactions on Circuits and Systems for Video Technology 25(5):892-896.

[6] Shen L., Zhang Z., Liu Z. (2014). Effective CU size decision for HEVC intra coding. IEEE Transactions on Image Processing 23(10):4232-4241.

[7] Liu X., Liu Y., Wang P., Lai C., Chao H. (2016). An adaptive mode Decision algorithm based on video texture characteristics for HEVC intra prediction. IEEE Transactions on Circuits Systems for Video Technology, 27(8): 1737-1748.

[8] Zhang T., Sun M-T., Zhao D., Gao W. (2017). Fast intra mode and CU size decision for HEVC. IEEE Transactions on Circuits Systems for Video Technology 27(8): 1714-1726.

[9] Mallikarachchi T., Fernando A., Arachchi H. (2014). Efficient coding unit size selection based on texture analysis for HEVC intra prediction. Paper presented at IEEE International Conference on Multimedia and Expo (ICME), Chengdu, China.

[10] Na S., Lee W., Yoo K. (2014). Edge-based fast mode decision algorithm for intra prediction in HEVC. Paper presented at IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA.

[11] Khan MUK., Shafique M., Grellert M., Henkel J. (2013). Hardware-software collaborative complexity reduction scheme for the emerging HEVC intra encoder. Paper presented at Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, France.

[12] Ye T., Zhang D., Dai F., Zhang Y. (2013). Fast mode decision algorithm for intra prediction in HEVC. Paper presented at the Fifth International Conference on Internet Multimedia Computing and Service (ICIMCS), Huangshan, Anhui, China.

[13] Shen L., Zhang Z., Liu Z. (2014). Effective CU size decision for HEVC intra coding. IEEE Transactions on Image Processing 23(10):4232.

[14] Shen L., Zhang Z., An P. (2013). Fast CU size decision and mode decision algorithm for HEVC intra coding. IEEE Transactions on Consumer Electronics 59(1):207.

[15] Lim K., Lee J., Kim S., Lee S. (2015). Fast PU skip and split termination algorithm for HEVC intra. IEEE Transactions on Circuits Systems for Video Technology 25(8).

[16] Shang X., Wang G., Fan T., Li Y. (2015). Fast CU size decision and PU mode decision algorithm in HEVC intra coding. Paper presented at IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada.

[17] Zhang Q., Wang X., Huang X., Su R. and Gan Y. (2015). Fast mode decision algorithm for 3D-HEVC encoding optimization based on depth information. Digital Signal Processing, 44:37-46.

[18] Zhang H., Ma Z. (2014). Fast Intra mode decision for high efficiency video coding (HEVC). IEEE Transactions on Circuits Systems for Video Technology 24(4):660–668.

[19] Cho S., Kim M. (2013). Fast CU splitting and pruning for suboptimal CU partitioning in HEVC intra coding. IEEE Transactions on Circuits Systems for Video Technology 23(9):1555–1564.

[20] Khan MUK., Shafique M., Grellert M., Henkel J. (2013). Hardware-software collaborative complexity reduction scheme for the emerging HEVC intra encoder. Paper presented at Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, France.

[21] Wang H-M., Lin J-K., Yang J-F. (2006). Fast inter mode decision based on hierarchical homogeneous detection and cost analysis for h.264/AVC coders. Paper presented at IEEE International Conference on Multimedia and Expo, Toronto, Ontario, Canada.

[22] Shen L., Liu Z., Zhang Z., Shi X. (2008). Fast Inter Mode Decision Using Spatial Property of Motion Field. IEEE Transactions on Multimedia 10(6):1208–1214.

[23] Goswami K., Lee J-H., Jang K-S., Kim B-G., Kwon K-K. (2014.) Entropy difference-based early skip detection technique for high efficiency video coding. Journal of Real-Time Image Processing.

[24] Wu D., Pan F., Lim K., Wu S., Li Z., Lin X., Rahardja S., Ko C. (2005). Fast intermode decision in h.264/AVC video coding. IEEE Transactions on Circuits and Systems for Video Technology 15(7):953–958.

[25] Shen L., Liu Z, Liu S., Zhang Z., An P. (2009). Selective disparity estimation and variable size motion estimation based on motion homogeneity for multi-view coding. IEEE Transactions on Broadcasting 55(4).

[26] Shen L., Liu Z., Yan T., Zhang Z., An P. (2010). View-adaptive motion estimation and disparity estimation for low complexity multiview video coding. IEEE Transactions on Circuits and Systems for Video Technology 20(6).

[27] Lee J-H., Park C-S., Kim B-G., Jun D-S., Jung S-H., Choi JS. (2013). Novel fast PU decision algorithm for the HEVC video standard. Paper presented at IEEE International Conference on Image Processing (ICIP), Melbourne, VIC, Australia.

[28] Alcocer E., Gutierrez R., Lopez-Granado O., Malumbres MP. (2016). Design and implementation of an efficient hardware integer motion estimator for an HEVC video encoder. Journal of Real-Time Image Processing 16(2): 547–557.

[29] Pastuszak G., Trochimiuk M. (2015). Algorithm and architecture design of the motion estimation for the H.265/HEVC 4K-UHD encoder. Journal of Real-Time Image Processing 12(2): 517–529.

[30] Mejia-Ocana A., de Frutos-Lopez M., Sanz-Rodriguez S., del Ama-Esteban O., Pelaez-Moreno C., Diaz-de Maria F. (2011). Low-complexity motion-based saliency map estimation for perceptual video coding. Paper presented at National Conference on Telecommunications (CONATEL), Catolica San Pablo, Arequipa, Peru.

[31] Richardson, I. E. (2003). H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia. Chichester, UK: Wiley.

[32] Smith, S. W. (1997). The Scientist and Engineer's Guide to Digital Signal Processing (2ª Ed.). California, USA: California Technical Publishing.

[33] Bossen F. (2012). Common test conditions and software reference configurations. JCT-VC Document, JCTVC-K1100.

[34] NVIDIA. (2018). CUDA Zone. Recuperado de https://www.nvidia.es/object/cuda-parallel-computing-es.html

[35] Khronos Group. (2018). OpenCL Overview. The open standard for parallel programming of heterogeneous systems. Recuperado de https://www.khronos.org/opencl/

[36] Jiang W., Ma H., Chen Y. (2012). Gradient based fast mode decision algorithm for intra prediction in HEVC. Paper presented at 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet), Yichang, China.

[37] Lake A. (2018). Intel SDK for OpenCL Applications. Getting the Most from OpenCL 1.2: How to Increase Performance by Minimizing Buffer Copies on Intel® Processor Graphics. Recuperado de https://software.intel.com/en-us/articles/getting-the-most-from-opencl-12-how-to-increase-performance-by-minimizing-buffer-copies-on-intel-processor-graphics

[38] Minoo K., Nguyen T. (2005). Perceptual video coding with H.264. Paper presented at Thirty-Ninth Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA.

[39] Gaurav R., Ates H.F. (2016) Efficient Quality of Multimedia Experience Using Perceptual Quality Metrics. Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics. Smart Innovation, Systems and Technologies 43:487-496. Springer, New Delhi.

[40] Faroudja Y. (2014). Improving Video Streaming and File Compression Efficiency without Affecting Quality. Paper presented at SMPTE Annual Technical Conference & Exhibition, Hollywood, CA, USA.

[41] Khronos Group. (2016). Online documentation. Recuperado de https://www.khronos.org/registry/OpenCL/sdk/1.1/docs/man/xhtml/sampler_t.html

[42] Intel Corporation. (2014). Introduction to Advance Motion Extension for OpenCL. Recuperado de https://software.intel.com/en-us/articles/intro-to-advanced-motion-estimation-extension-for-opencl

[43] Intel Corporation. (2012). Intel SDK for OpenCL Applications. Performance Interactions of OpenCL Code and Intel Quick Sync Video on Intel HD Graphics 4000. Recuperado de https://software.intel.com/en-us/articles/performance-interactions-of-opencl-code-and-intel-quick-sync-video-on-intel-hd-graphics-4000

[44] Khronos Group. (2016). Online documentation. Recuperado de https://www.khronos.org/registry/cl/extensions/intel/cl_intel_advanced_motion_estimation.txt

[45] Fernández D. G., Del Barrio A. A., Botella G., García C. (2018). HEVC optimization based on human perception for real-time environments. Multimedia Tools and Applications.

[46] Fernández D. G., Del Barrio A. A., Botella G., García C. (2018). Fast and effective CU size decision based on spatial and temporal homogeneity detection. Multimedia Tools and Applications, 77(5):5907–5927.

[47] Fernández D. G., Del Barrio A. A., Botella G., García C., Prieto M., Hermida R. Complexity reduction in the HEVC/H265 standard based on smooth region classification. Digital Signal Processing 73:24-39.

[48] Fernández D. G., Del Barrio A. A., Botella G., Meyer-Baese U., Meyer-Baese A., Grecos C. (2017). Information fusion based techniques for HEVC. Proceedings 10223 SPIE Real-Time Image and Video Processing. Paper presented at SPIE Commercial + Scientific Sensing and Imaging, Anaheim, California, United States.

[49] Fernández D. G., Del Barrio A. A., Botella G., Meyer-Baese U., Meyer-Baese A., Grecos C. (2017). Pre-processing techniques to improve HEVC subjective quality. Proceedings 10223 SPIE Real-Time Image and Video Processing. Paper presented at SPIE Commercial + Scientific Sensing and Imaging, Anaheim, California, United States.

[50] Fernández D. G., Del Barrio A. A., Botella G., García C. (2016). Fast CU size decision based on temporal homogeneity detection. Paper presented at Conference on Design of Circuits and Integrated Systems (DCIS). IEEE.

[51] Fernández D. G., Del Barrio A. A., Botella G., Garcia C., Meyer-Baese U., Meyer-Baese A. (2016). HEVC optimizations for medical environments. Proceedings 9871 SPIE Sensing and Analysis Technologies for Biomedical and Cognitive Applications. Paper presented at SPIE Commercial + Scientific Sensing and Imaging, Baltimore, Maryland, United States.

[52] Fernández D. G., Del Barrio A. A., Botella G., Garcia C. (2016). 4K-based intra and inter prediction techniques for HEVC. Proceedings 9897 SPIE Real-Time Image and Video Processing. Paper presented at SPIE Photonics Europe, Brussels, Belgium.

[53] Leng J., Sun L., Ikenaga T., Sakaida S. (2011). Content Based Hierarchical Fast Coding Unit Decision Algorithm For HEVC. Paper presented at International Conference on Multimedia and Signal Processing, Guilin, Guangxi, China.

[54] Flynn D., Marpe D., Naccari M., Nguyen T., C. Rosewarne, K. Sharman, J. Sole, J. Xu. (2016). Overview of the Range Extensions for the HEVC Standard: Tools, Profiles and Performance. IEEE Transactions on Circuits and Systems for Video Technology 26(1).

[55] Zhou M., Gao W., Jiang M., Yu H. (2012). HEVC Lossless Coding and Improvements. IEEE Transactions on Circuits and Systems for Video Technology 22(12).

[56] The Cancer Imaging Archive. (2016). Recuperado de http://www.cancerimagingarchive.net/

[57] MicroDicom Application. (2016). Recuperado de http://www.microdicom.com/

[58] FFmpeg multimedia framework. (2016). Recuperado de https://www.ffmpeg.org/

[59] Shen L., Liu Z., Zhang X., Zhao W., Zhang Z. (2013). An effective CU size decision method for HEVC encoders. IEEE Transactions on Multimedia 15(2):465–470.

[60] Xiong J., Li H., Wu Q., Meng F. (2014). A fast HEVC inter CU selection method based on pyramid motion divergence. IEEE Transactions on Multimedia 16(2):559–564.

[61] Xiong J, Li H, Meng F, Zhu S, Wu Q, Zeng B (2014b) MRF-based fast HEVC inter CU decision with the variance of absolute differences. IEEE Trans Multimedia 16(8):2141–2153.

[62] Ahn S., Lee B., Kim M. (2015). A novel fast CU encoding scheme based on spatio-temporal encoding parameters for HEVC inter coding IEEE Transactions on Circuits and Systems for Video Technology 25(3):422–435.

[63] Xiong J, Li H, Meng F, Wu Q, Ngan KN (2015) Fast HEVC inter CU decision based on latent SAD estimation. IEEE Transactions on Multimedia 17(12):2147–2159.

[64] Ramezanpour M., Zargari F. (2016). Fast HEVC I-frame coding based on strength of dominant direction of CUs. Journal of Real-Time Image Processing 12(2): 397–406.

Capítulo 6

Contribuciones. Captura de vídeo

6.1 Introducción

En apartados anteriores se han presentado una serie de algoritmos que han sido verificados mediante entrada y salida de fichero. Esto supone que como entrada al codificador se ha utilizado un fichero de vídeo sin comprimir YUV 4:2:0 y se ha obtenido como salida del codificador un fichero que contiene el vídeo comprimido (extensión *.265 o *.hevc).

Para poder verificar el sistema de codificación en tiempo real se decidió implementar una gestión de la entrada de vídeo que permitiese introducir al sistema una señal de vídeo digital sin comprimir. Dentro de los diferentes interfaces de entrada/salida de vídeo y audio existentes, se optó por el uso de *Serial Digital Interface (SDI)* al ser el interfaz más ampliamente difundido dentro del mercado de broadcast profesional. Por lo tanto, se requerirá un dispositivo capaz de extraer el vídeo del interfaz SDI y proporcionárselo al procesador encargado de realizar la codificación.

Habiendo centrado la implementación de ciertos algoritmos de optimización de la codificación HEVC en GPU embebidas de Intel, la comunicación entre el dispositivo encargado de gestionar la entrada de vídeo y el microprocesador/GPU a cargo de la codificación puede gestionarse fácilmente a través del bus *Peripheral Component Interconnect Express (PCIe)*. De esta forma, la tarjeta hardware encargada de realizar la captura de vídeo se gestiona como un periférico PCIe.

En los siguientes apartados se describirán con más detalle los interfaces utilizados, el ancho de banda requerido para trabajar en el sistema de codificación y ciertos detalles

relativos a la implementación del sistema, así como la arquitectura hardware finalmente escogida.

6.2 Conceptos previos

6.2.1 PCIe

6.2.1.1 Topología

La tecnología PCIe proporciona una solución para el transporte de datos que permite la utilización de altas tasas de transferencia. PCIe es un protocolo de interconexión serie punto a punto que permite establecer una comunicación fiable y escalable. Para la transferencia de datos se utilizan dos líneas diferenciales independientes, una para la transmisión y otra para la recepción. Este conjunto de 4 pines es conocido como *lane*. Además, se requiere el uso de un pin de reset y un reloj. Se trata de una solución escalable ya que aumentando en número de lanes (x1, x2, x4, x8, x16) se consigue aumentar la tasa de transferencia.

PCIe se basa en un protocolo basado en transacciones mediante paquetes de datos. Dentro de los paquetes de datos que se manejan existen accesos a memoria, a periféricos (I/O), datos de configuración, señalización de interrupciones, confirmación de recepción de datos, etc. Un sistema PCIe puede visualizarse con una estructura similar a una red IP, donde cada dispositivo es conectado a la red por medio de un switch. La comunicación entre los distintos dispositivos se realiza por medio de paquetes que incluyen control de flujo, detección de errores y retransmisiones. Para reconocer a cada dispositivo físico se utiliza un identificador. En la Figura 6.1, se muestra un ejemplo básico de arquitectura PCIe.

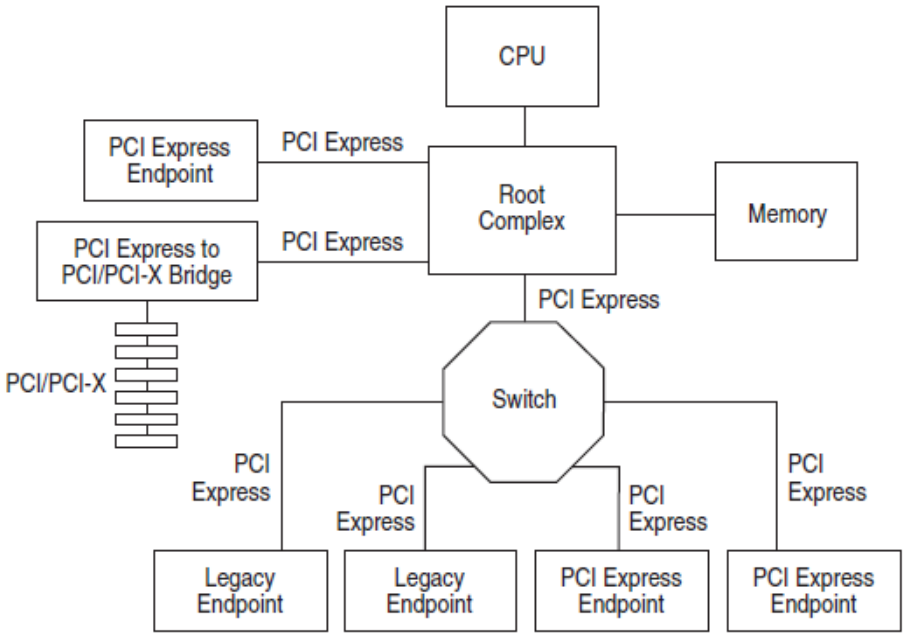


Figura 6.1. Topología PCIe.
Extraído de [1].

El dispositivo que aparece como *Root Complex* es el encargado de conectar la CPU y la memoria con el bus local PCIe, generando las peticiones por parte de la CPU. En la topología mostrada en la Figura 6.1, Root Complex posee varios puertos que le permiten conectarse a un switch y a varios dispositivos PCIe directamente (*endpoint* y el conversor PCIe a PCI/PCI-X. Endpoint es un dispositivo capaz de gestionar transacciones PCIe para sí mismo o para un dispositivo que no sea compatible con PCIe. El switch simplemente permite conectar varios dispositivos PCIe al Root Complex.

Dentro de una arquitectura PCIe se definen tres capas lógicas: *Transaction Layer*, *Data Link Layer* y *Physical Layer*. Cada capa a su vez se divide en transmisión y recepción. En la Figura 6.2, se muestra un diagrama con las diferentes capas.

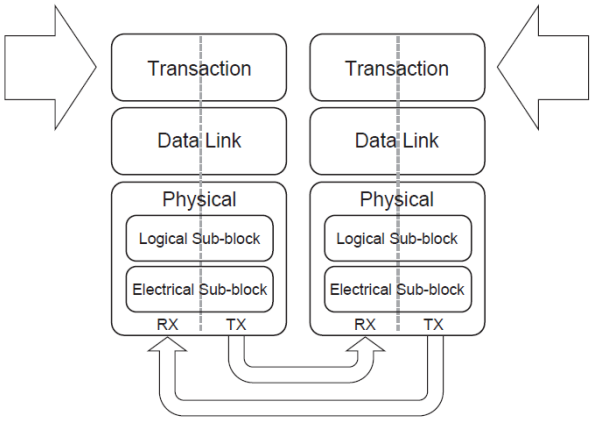


Figura 6.2. Diferentes capas lógicas dentro de PCIe.
Extraído de [1].

Las transacciones de datos dentro de un sistema PCIe se llevan a cabo por medio de un intercambio de paquetes. Los paquetes son elaborados por la capas Transaction y Data Link. Como los paquetes fluyen a través de todas las capas, la información que contienen es extendida para que puedan procesarse en cada capa. Desde el punto de vista del receptor, los paquetes se transforman desde la capa Physical, atravesando la capa Data Link hasta alcanzar la capa Transaction, donde se procesarán los paquetes de tipo *Transaction Layer Packets (TLP).* En la Figura 6.3, se muestra el flujo de paquetes a través de las diferentes capas.

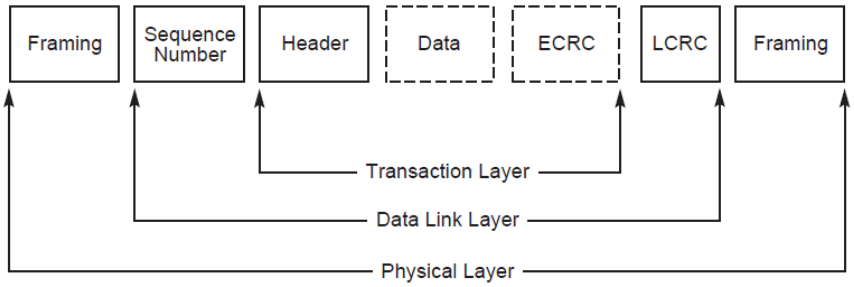


Figura 6.3. Flujo de paquetes a través de las diferentes capas lógicas.
Extraído de [1].

6.2.1.2 Transaction Layer Packets (TLP)

Los TLP son utilizados para realizar las transacciones relativas a lecturas y escrituras, y algunos tipos de eventos. Describiendo las capas de una manera muy simple, la capa Physical incluye toda la circuitería necesaria para el funcionamiento del interfaz, incluyendo el *driver*, los búferes de entrada, conversiones serie a paralelo y viceversa, PLL (*phase lock loop*) para la generación de relojes y circuitos para el ajuste de impedancias. Por lo tanto, la capa Physical convierte toda la información recibida por la capa Data Link en un formato adecuado para su transmisión a través de un enlace PCIe, empleando la frecuencia y el número de lanes que sean compatibles con el dispositivo PCIe con el que se ha establecido la comunicación. En la recepción, la capa Physical proporciona los datos recibidos por el interfaz físico a la capa Data Link para su procesado.

La capa Data Link sería la responsable de asegurar que cada TLP alcanza su destino correctamente por medio de integridad de datos y detección y corrección de errores. Para ello, anexa al TLP cabeceras extra e implementa un mecanismo de gestión de errores basado en *Cyclic Redundancy Check (CRC)*, asegurando la integridad del TLP tras su transmisión. Existe un mecanismo de retransmisión de paquetes basado en paquetes de respuesta que indican que el paquete se ha recibido correctamente por parte del receptor, de esta forma se asegura que no se han perdido TLP durante la transmisión. El control de flujo asegura que cada paquete se envía únicamente cuando un receptor está listo para recibirlo. Desde el punto de vista de un usuario de PCIe, los paquetes siempre deberían ser recibidos, en caso contrario podemos asumir un mal funcionamiento del bus.

La capa Transaction es la encargada de conformar los TLP para su transmisión y de interpretar los TLP recibidos. El formato del TLP soporta diferentes tipos de direccionamiento dependiendo del espacio de memoria al que se acceda. Los espacios de memoria que se manejan en PCIe son *Memory*, *I/O*, *Configuration* y *Message*; habiendo tipos de transacción distintos para cada espacio. Memory e I/O utilizan los tipos de transacción de lectura y escritura, difieren en que en el primer caso se accede a una posición situada en la memoria del sistema y en el segundo, una posición situada en el espacio de memoria de los periféricos. En el espacio de memoria Configuration se permiten igualmente lecturas y escrituras, pero con el objetivo de realizar la configuración del dispositivo. El espacio Message, utiliza tipos de transacción dependientes del fabricante y las transacciones se utilizan para señalizar eventos o para utilizar mensajes de propósito general. Por otro lado, dentro de la especificación PCIe se permite utilizar direccionamiento de 32 o de 64bits, lo que conlleva la modificación de la estructura de los paquetes. La especificación PCIe determina que si la memoria asociada al dispositivo es inferior a 4GB se debe utilizar direccionamiento de 32bits. Los TLP deben estar alineados a 4 bytes y las direcciones se incrementan en *Double Words (DW)* de 4 bytes. Las transacciones son realizadas mediante peticiones (*requests*) y eventos de terminación de la transacción (*completions*). Las terminaciones son utilizadas únicamente en el caso de requerirse, como pueden ser los datos leídos, y una señal de consecución de escritura a un periférico o al espacio de memoria de configuración. Mediante el campo que indica el identificador de la transacción denominado Tr*ansaction ID* de la cabecera del paquete de terminación, se asocia dicho paquete al correspondiente paquete de petición. En la Figura 6.4, se muestra una representación de la cabecera asociada a un paquete de petición utilizando un direccionamiento de 32 bits. En la Figura 6.5, se muestra el formato correspondiente a un paquete de terminación.

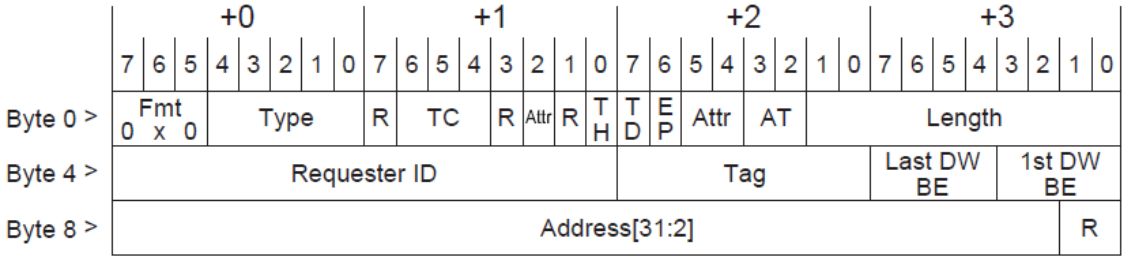


Figura 6.4. Formato de un paquete de petición con direccionamiento de 32 bits. Extraído de [1].

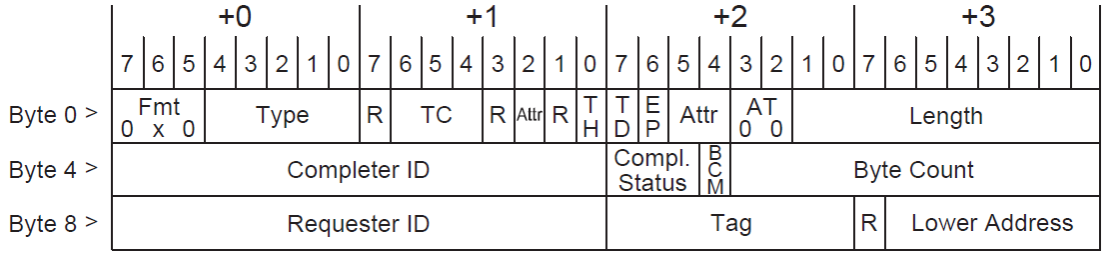


Figura 6.5. Formato de un paquete de terminación. Extraído de [1].

A continuación, se describen los campos más relevantes que aparecen en las Figuras 6.4 y 6.5 para la aplicación desarrollada, ya que no es objetivo de este trabajo una descripción en detalle de la arquitectura PCIe. Para una descripción en profundidad se debe consultar la especificación PCIe definida en [1].

- *Fmt*. Formato del TLP indicando el número de DW utilizados:
  - 000b→3 DW de cabecera, sin datos;
  - 001n→4 DW de cabecera, sin datos;
  - 010b→3 DW de cabecera, con datos;
  - 011b→4 DW de cabecera, con datos;
  - 00b→ Prefijo del TLP.
- *Type*. Fmt y Type definen el tipo de paquete. Conforme a la Tabla 6.1.
- *TD*. Se fuerza a 0 para indicar que no se usará CRC adicional al utilizado en la capa Data Link. No tiene sentido sobrecargar el sistema si se confía en que el hardware funciona correctamente.
- *Length*. Longitud del payload de datos indicado en DW.
- *Completer ID*. Identifica el dispositivo que informa de la terminación.
- *Completion Status*. Los diferentes valores se definen en la Tabla 6.2.
- *BCM*. Byte Count Modified. No utilizado en PCIe.
- *Byte Count*. La cantidad de bytes que quedan por transferir para la petición correspondiente.
- *Tag*. Junto al campo Requester ID conforman el Transaction ID.
- *Lower Address*. Byte más bajo de la dirección del byte de comienzo asociada a la terminación

Tabla 6.1. Tipos de paquetes en PCIe.
Extraída de [1].

TLP Type	Fmt [2:0][2] (b)	Type [4:0] (b)	Description
MRd	000 001	0 0000	Memory Read Request
MRdLk	000 001	0 0001	Memory Read Request-Locked
MWr	010 011	0 0000	Memory Write Request
IORd	000	0 0010	I/O Read Request
IOWr	010	0 0010	I/O Write Request
CfgRd0	000	0 0100	Configuration Read Type 0
CfgWr0	010	0 0100	Configuration Write Type 0
CfgRd1	000	0 0101	Configuration Read Type 1
CfgWr1	010	0 0101	Configuration Write Type 1
TCfgRd	000	1 1011	Deprecated TLP Type[3]
TCfgWr	010	1 1011	Deprecated TLP Type[3]
Msg	001	1 0$r_2r_1r_0$	Message Request – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-18).
MsgD	011	1 0$r_2r_1r_0$	Message Request with data payload – The sub-field r[2:0] specifies the Message routing mechanism (see Table 2-18).
Cpl	000	0 1010	Completion without Data – Used for I/O and Configuration Write Completions with any Completion Status. Also used for AtomicOp Completions and Read Completions (I/O, Configuration, or Memory) with Completion Status other than Successful Completion.
CplD	010	0 1010	Completion with Data – Used for Memory, I/O, and Configuration Read Completions. Also used for AtomicOp Completions.
CplLk	000	0 1011	Completion for Locked Memory Read without Data – Used only in error case.
CplDLk	010	0 1011	Completion for Locked Memory Read – otherwise like CplD.

Tabla 6.2. Valores del campo Completion Status.
Extraída de [1].

Completion Status[2:0] Field Value (b)	Completion Status
000	Successful Completion (SC)
001	Unsupported Request (UR)
010	Configuration Request Retry Status (CRS)
100	Completer Abort (CA)
all others	Reserved

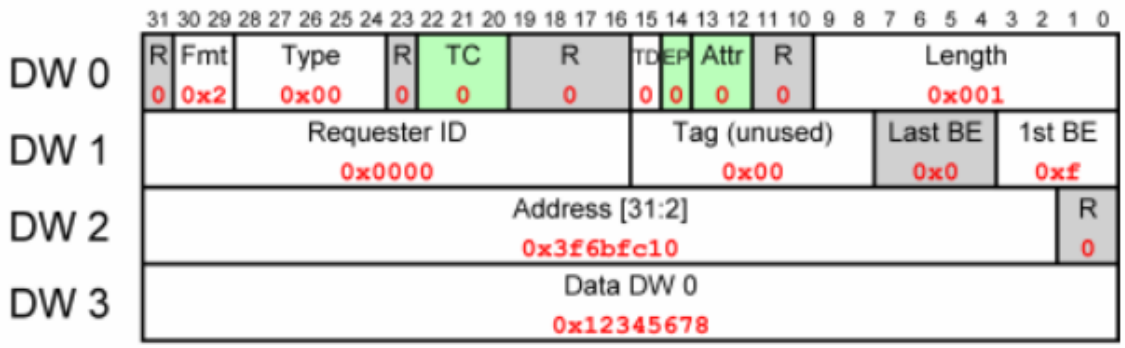


Figura 6.6. Ejemplo de TLP para petición de escritura.
Extraído de [2].

En la Figura 6.6 se muestra un paquete que se corresponde con una simple escritura de un dato de 32bits utilizando direccionamiento de 32bits. En la Figura 6.6, los campos en gris son campos reservados, lo que implica que el emisor los debe de poner a 0. Los campos en verde pueden contener valores distintos de cero pero no serán utilizados por la aplicación diseñada. A continuación, se describirán los campos utilizados:

- *Length* = 1. Indica que el TLP incluye únicamente un DW de datos (32 bits).
- *Requester ID* = 0. Indica que la petición es realizada desde el root complex.
- *1st BE field (1st Double-Word Byte Enable).* Indica qué bytes dentro del primer DW son válidos para ser escritos.
- *Last BE field*. Indica cuál es el último byte dentro del último DW. Si Length es 1 debe de ser 0, al coincidir último y primer DW.
- *Address*. Dirección de escritura del dato.
- *Data DW 0*. Dato a escribir.

En la Figura 6.7 se muestra un ejemplo del formato que sigue un TLP para la petición de una lectura por parte de la CPU a un periférico. Una lectura de una posición de memoria siempre implica dos TLP. Un TLP es la petición propiamente dicha (request), y otro TLP es el dato leído (completion). En la Figura 6.7, se muestra un ejemplo de la lectura de un DW de una dirección concreta. Como se puede apreciar para realizar dicha lectura se requieren 3 DW.

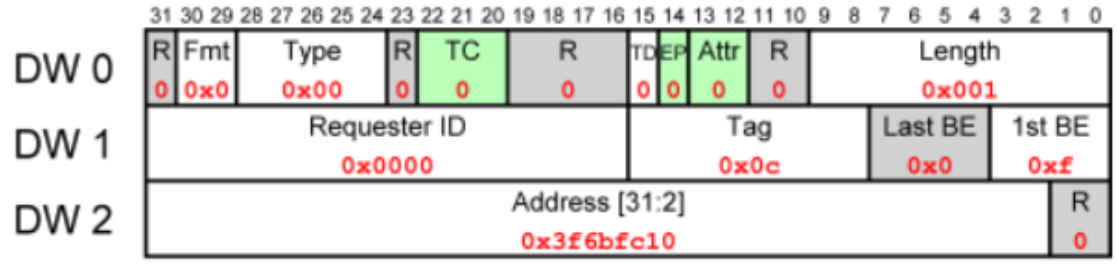


Figura 6.7. Ejemplo de TLP para petición de lectura.
Extraído de [2].

El TLP de lectura es bastante similar al TLP de escritura, pero destacan las siguientes diferencias:

- El campo *Fmt/Type* se modifica para indicar que es una petición de lectura.
- El *Requester ID*, en la escritura es un campo obligatorio pero que en la práctica no es utilizado. En el caso de la lectura, es un campo crucial ya que indica al receptor del paquete a qué dispositivo enviar su respuesta.
- El campo *Tag* en sí mismo no configura nada, pero en las lecturas es utilizado como un identificador para realizar un seguimiento del paquete. El paquete respuesta a la petición de lectura copia el valor del Tag, de esta forma el dispositivo que realizó la petición de lectura puede asociar los paquetes de respuesta a la petición correspondiente.

En la Figura 6.8 se muestra un ejemplo del formato que sigue el TLP de respuesta para la petición de una lectura indicada en el TLP de la Figura 6.7.

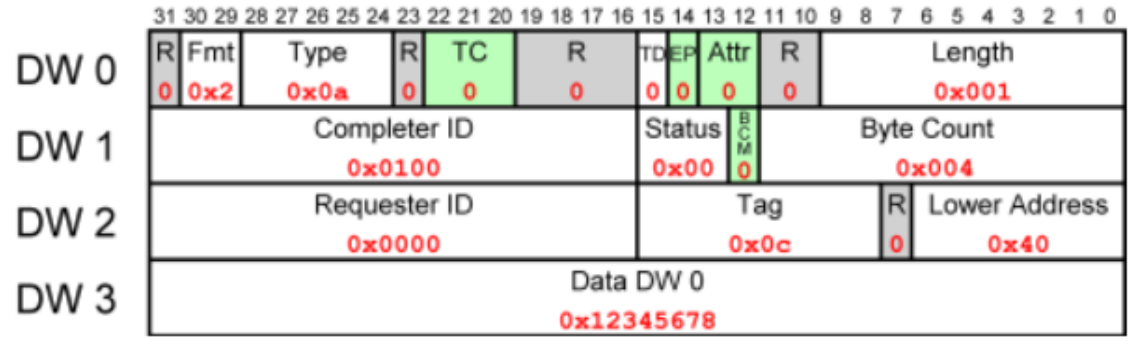


Figura 6.8. Ejemplo de TLP de respuesta a una petición de lectura.
Extraído de [2].

Para cada petición de lectura, el equipo destinatario de la petición debe de generar una respuesta aunque no haya podido generarla con éxito. Hay que tener en cuenta ciertos aspectos, como el máximo tamaño de TLP que soporta un dispositivo. Si la petición supera el tamaño máximo de TLP que soporta el dispositivo que recibe la petición, se generarán varios TLP de respuesta. Dicho parámetro es conocido como *Max_Payload_Size* cuyos valores permitidos son 128, 256, 512, 1024, 2048 y 4096 bytes. Por otro lado, las peticiones de lectura no pueden superar el tamaño impuesto por el parámetro *Max_Read_Request_Size* cuyos valores permitidos son los mismos que los definidos para Max_Payload_Size.

Es importante resaltar que las peticiones de escritura/lectura de memoria no pueden realizar accesos de memoria que crucen fronteras de 4 KB (*4 KB-boundary*), tal y como se detalla en el apartado 2.2.7 de la especificación PCIe [1].

En la Figura 6.8, es dispositivo con ID 0x0100 responde a la petición generada por medio del TLP de la Figura 6.7, indicando en el campo TAG la asociación entre ambos TLP. A continuación, se profundiza en la información que contienen los campos más relevantes de un TLP de respuesta:

- El campo *Fmt/Type* se modifica para indicar que es una respuesta a una petición de lectura.
- *Length field*, indica el número de DW de datos que contiene el TLP.
- Status es 0, lo que indica que la lectura se llevó a cabo con éxito.
- *Byte Count*, indica el número de bytes que quedan por transmitir, incluyendo los asociados al paquete actual. Indispensable para gestionar respuestas que implican varios TLP.
- *Tag*, como se ha comentado debe de ser una copia del campo Tag de la petición correspondiente.
- *Lower address*, indica los 7 bits menos significativos de la dirección desde la cual el primer byte de este TLP fue leído.
- *Data DW*, es el dato leído.

6.2.1.3 Interrupciones

PCIe soporta dos tipos de interrupciones: *Legacy INTx* y *Message Signaled Interrupt (MSI* y *MSI-X)*. INTx está relevado a aquellos casos en los que no se soporte MSI, por lo que tiene una labor de compatibilidad con dispositivos antiguos como pueden ser puentes PCIe/PCI (-X). Como la aplicación que se ha diseñado implementa un endpoint PCIe, se utilizarán interrupciones del tipo MSI tal y como recomienda la especificación PCIe.

Tanto en MSI como en MSI-X, las interrupciones son generadas mediante paquetes de escritura en una posición específica de la memoria del host. El host a través del sistema operativo, llamará a la rutina de atención a la interrupción cuando detecte una escritura en dicha posición de memoria. En MSI se utiliza una única dirección y un dato de valor variable, siendo dicho valor el vector de interrupción que indica el motivo de la interrupción. MSI-X utiliza una tabla de direcciones independientes y parejas de datos para cada vector de interrupción. Asignando posiciones de memoria específicas para cada tipo de interrupción, el host determina qué dispositivo generó la interrupción y la causa de interrupción.

6.2.1.4 Espacio de memoria de configuración

Los dispositivos PCI disponen de un conjunto de registros de configuración conocidos como espacio de configuración (*configuration space*) y PCIe introduce un espacio de configuración avanzando (*extended configuration space*). Dichos registros se encuentran mapeados en la memoria del sistema. El driver del dispositivo PCIe y el software de diagnóstico deben acceder al espacio de configuración por medio de un conjunto de rutinas definidas por un interfaz de programación de aplicación (*application programming interface (API)*) proporcionado por el sistema operativo.

El proceso de configuración de un dispositivo PCIe no se detallará debido a que está fuera del ámbito del trabajo realizado, si se desea, se puede consultar [1] para acceder a la

información relativa a este proceso. Aun así, es importante conocer algunos de los registros cuyo valor es relevante para la aplicación desarrollada:

- *Vendor ID*: Identifica el fabricante del dispositivo. Los identificadores válidos son asignados por *PCI Special Interest Group (PCI-SIG)* para garantizar que cada identificador es único.
- *Device ID:* Identifica el dispositivo en cuestión. Los valores son asignados por el fabricante del dispositivo.
- *Revision ID.* Identifica futuras revisiones del dispositivo PCIe. Puede ser utilizado para identificar varios diseños realizados sobre el dispositivo PCIe.
- *Subsystem Vendor ID*: Identifica el fabricante del subsistema que usa el dispositivo. Por ejemplo, en el caso de usar una FPGA de Xilinx, el Vendor ID identifica a Xilinx y el Subsystem Vendor ID al fabricante de la tarjeta que incorpora la FPGA.
- *Subsystem ID*: Identifica al subsistema en cuestión.
- *Class Code*. Identifica la función general que desempeña un dispositivo, por ejemplo controlador de memoria.

Adicionalmente, en el espacio de memoria de configuración también residen las capacidades del dispositivo, donde se indica el tamaño del BAR, el soporte a interrupciones de tipo MSI/MSI-X, el número de lanes dedicadas al enlace de comunicación, velocidad del enlace y los parámetros Max_Payload_Size y Max_Read_Request_Size. Estas capacidades sirven para preestablecer los parámetros necesarios en la comunicación entre el Root Complex y el endpoint PCIe, ya que si uno de los dispositivos no soporta alguna de las capacidades del otro, el enlace de comunicación se establecerá conforme a las capacidades más restrictivas. Por ejemplo, aunque el endpoint PCIe soporte 16 lanes si el Root Complex sólo soporta 4 lanes, el enlace de comunicación se establecerá conforme al mínimo que en este caso es de 4 lanes.

6.2.1.5 Base Address Registers (BAR)

Para cada dispositivo endpoint PCIe existe un conjunto de registros de direcciones base que es utilizado principalmente para dos propósitos. Primero, durante la inicialización sirven como mecanismo para que el dispositivo solicite un espacio de memoria dentro de la memoria de sistema. Y segundo, tras realizarse la inicialización, la BIOS o el sistema operativo determina qué direcciones asignar al dispositivo y los BAR son programados con las direcciones que el dispositivo utilizará para realizar una descodificación de direcciones. De esta manera, se puede acceder desde el host al dispositivo con las direcciones asignadas por la BIOS o el sistema operativo.

6.2.1.6 Tasa de transferencia

La tasa de transferencia máxima especificada por la generación 1 (Gen1) de sistemas basados en PCIe es 2.5 Gb/s, alcanzando un máximo de 5 Gb/s en la generación 2 (Gen2) y hasta 8 Gb/s en la generación 3 (Gen3). Estas tasas son valores máximos ideales para cada lane en una única dirección, no siendo realmente la tasa a la cual los datos son transferidos a través del sistema. La tasa de transferencia efectiva es menor debido a la sobrecarga del bus y a otros aspectos que se resumirán a continuación.

Tanto los protocolos basados en Gen1 como los basados en Gen2 utilizan un esquema de codificación 8b/10b, por el que a cada dato de 8 bits se le asigna un símbolo de 10bits,

con el objetivo de conseguir una señal DC balanceada (la amplitud media de la señal es 0 por lo que no tiene componente DC) pero permitiendo suficientes cambios de estado que posibiliten al equipo receptor recuperar el reloj por medio de los datos recibidos.
Por lo tanto, el ancho de banda teórico puede calcularse tal y como se indica en la Ecuación 6.1. Los 2.5 Gb/s son multiplicados por dos debido a los dos sentidos (transmisión y recepción) existentes en una comunicación PCIe.

$$Ancho\ de\ banda\ PCIe\ (bytes) = \frac{2.5\frac{Gb}{s} * 2 * Lane\ Width}{10bits/byte}. \quad (6.1)$$

Debido a que son transmitidos 10bits para cada byte de datos útiles, existe una pérdida asociada a la codificación 8b/10b de un 20%. En Gen3 se utiliza una codificación 128b/130b, de forma que se reduce la pérdida a tan sólo un 2%. De esta forma, el ancho de banda teórico asociado a Gen1 pasa a ser 2 Gb/s para una única dirección y lane, en lugar de los 2.5 Gb/s. De manera similar, Gen2 puede proporcionar teóricamente 4 Gb/s y Gen3 hasta 7.9 Gb/s.

Como PCIe basa las comunicaciones en el uso de paquetes de datos, existe una sobrecarga del bus debido al uso de las cabeceras necesarias para conformar los paquetes. Las capas Transaction, Data Link y Physical añaden datos sobre cada TLP efectivo añadiendo una sobrecarga al bus que puede llegar hasta suponer ≈ 1 Gb/s [2]. Utilizar tamaños de payload elevados permite reducir dicha sobrecarga, ya que la proporción de número de cabeceras respecto a los datos efectivos es reducida.

6.2.1.7 Bus Master (DMA)

En el contexto de PCIe el término Bus Máster se refiere a la habilidad de un puerto PCIe de iniciar las transacciones. Su aplicación más común es la implementación de un dispositivo de acceso directo a memoria (*Direct Memory Access*, *DMA*). Con este tipo de implementación el dispositivo puede acceder a la memoria del sistema sin necesidad de la intervención de la CPU.

Las transacciones de datos de entrada/salida programadas (*Programmed Input/Output*, *PIO*) son ejecutadas directamente por orden de la CPU y están limitadas normalmente a uno o dos DW al mismo tiempo. Para transferencias de grandes cantidades de datos las implementaciones basadas en DMA son la solución idónea, ya que proporcionan elevadas tasas de transferencia por no estar limitado el número de DW a transferir y además no se requiere el uso de la CPU, lo que permite liberar la carga computacional asociada al movimiento de datos.

Las implementaciones *Bus Master DMA (BMD)* permiten que un dispositivo endpoint PCIe inicie los accesos a la memoria del sistema (lecturas y escrituras) por sí mismo. De esta forma, la CPU no tiene que iniciar el movimiento de datos, limitándose a conceder permisos de acceso a ciertas ventanas de memoria donde se situarán los búferes a los que accederá el dispositivo BMD y a tareas de control a determinar desde el driver. La comunicación con el dispositivo se establece a través de los BAR.

6.2.2 SDI

6.2.2.1 Introducción

Los requerimientos técnicos para el trasporte de señales de vídeo son descritos en diversos estándares creados por T*he Society of Motion Picture and Television Engineers (SMPTE)*. El transporte de este tipo de señales sobre canales de transmisión serie es conocido como SDI.

Las ventajas que aporta este interfaz de comunicación han permitido su amplia difusión entre los diferentes fabricantes de equipos. Entre sus ventajas destacan la facilidad de uso, un interfaz eléctrico simple (*single-ended signaling*, un cable conectado a un voltaje variable y otro a tierra) y la posibilidad de transmitir vídeo y audio digital sin comprimir en tiempo real. El principal objetivo perseguido por SMPTE, al igual que en otros casos de estandarización, es la interoperabilidad entre los distintos fabricantes. Para tal propósito, se han elaborado diversos estándares en función de la resolución del vídeo que se utilice en la comunicación. SMPTE 295M es el estándar utilizado para señales SD, SMPTE 292 en HD y SMPTE 424M en resoluciones Full HD que requieran un ancho de banda de 3 Gb/s.

6.2.2.2 Tasa de transferencia

El bitrate requerido por el interfaz SDI puede ser fácilmente calculado haciendo uso de la Ecuación 6.2.

$$Bitrate\ (bits/s) = N_l * N_s * N_b * N_f * 2. \quad (6.2)$$

Siendo N_l, el número de líneas por fotograma. N_s es el número de muestras de luminancia por línea. N_b representa el número de bits por muestra y N_f el número de fotogramas por segundo. Se asume que se trabaja con el muestreo de color 4:2:2, por ello se realiza la multiplicación por 2 para obtener el número total de muestras (luminancia + crominancia).

En la Tabla 6.3 se indica el bitrate requerido para diferentes formatos de vídeo, así como otros parámetros relevantes como son el número de líneas de vídeo totales, el número de líneas de vídeo activas, número de muestras activas y totales por línea, etc. Para los estándares indicados, SDI utiliza muestreo de color 4:2:2 y 10 bits por píxel. Como se puede observar, los formatos 1920x1080/60-50-59.94P alcanzan un bitrate próximo a 3 Gb/s. Por ello, son denominados como 3G-SDI.

Tabla 6.3. Parámetros relevantes para diferentes formatos de vídeo.
Extraída de [4].

SPECIFICATION	SYSTEM NO.	SYSTEM NOMENCLATURE	LUMINANCE or R'G'B' SAMPLES per ACTIVE LINE ($LUMA_{ACTIVE}$)	ACTIVE LINES per FRAME	FRAME RATE (F_R) (Hz)	INTERFACE SAMPLING FREQUENCY fs (MHz)	LUMINANCE SAMPLE PERIODS per TOTAL LINE ($LUMA_{SAMPLE}$)	TOTAL LINES per FRAME (LNS_{FRAME})	BIT RATE(BIT_{RATE})	PATHOLOGICAL LENGTH ($PATH_{LENGTH}$)
SMPTE274M	1	1920 X 1080/60/P	1920	1080	60	148.5	2200	1125	2970000000	1.29E-05
	2	1920 X 1080/59.94/P	1920	1080	59.94	148.35	2200	1125	2967030000	1.29E-05
	3	1920 X 1080/50/P	1920	1080	50	148.5	2640	1125	2970000000	1.29E-05
	4	1920 X 1080/60/I	1920	1080	30	74.25	2200	1125	1485000000	2.59E-05
	5	1920 X 1080/59.94/I	1920	1080	29.97	74.18	2200	1125	1483515000	2.59E-05
	6	1920 X 1080/50/I	1920	1080	25	74.25	2640	1125	1485000000	2.59E-05
	7	1920 X 1080/30/P	1920	1080	30	74.25	2200	1125	1485000000	2.59E-05
	8	1920 X 1080/29.97/P	1920	1080	29.97	74.18	2200	1125	1483515000	2.59E-05
	9	1920 X 1080/25/P	1920	1080	25	74.25	2640	1125	1485000000	2.59E-05
	10	1920 X 1080/24/P	1920	1080	24	74.25	2750	1125	1485000000	2.59E-05
	11	1920 X 1080/23.98/P	1920	1080	23.98	74.18	2750	1125	1483762500	2.59E-05
SMPTE296M	1	1280 X 720/60	1280	720	60	74.25	1650	750	1485000000	1.72E-05
	2	1280 X 720/59.94	1280	720	59.94	74.18	1650	750	1483515000	1.73E-05
	3	1280 X 720/50	1280	720	50	74.25	1980	750	1485000000	1.72E-05
	4	1280 X 720/30	1280	720	30	74.25	3300	750	1485000000	1.72E-05
	5	1280 X 720/29.97	1280	720	29.97	74.18	3300	750	1483515000	1.73E-05
	6	1280 X 720/25	1280	720	25	74.25	3960	750	1485000000	1.72E-05
	7	1280 X 720/24	1280	720	24	74.25	4125	750	1485000000	1.72E-05
	8	1280 X 720/23.98	1280	720	23.98	74.18	4125	750	1483762500	1.73E-05

6.2.2.3 Sincronismos y datos auxiliares (audio)

La familia de estándares SDI elaborados por SMPTE son ampliamente utilizados en estudios y centros de producción de vídeo para transportar tanto la señal de vídeo sin aplicar compresión como el audio. El audio es tratado como datos auxiliares (*ancillary data*) insertándolo dentro de determinadas líneas de vídeo no activo. Como se puede apreciar en la Tabla 6.3, existe un gran número de líneas y de muestras dentro de una línea pertenecientes a vídeo no activo. El concepto "vídeo no activo" se refiere a que no contienen píxeles de la imagen. En la Figura 6.9 se muestra una representación del vídeo activo para resoluciones 1080p según el estándar SMPTE 274M.

El comienzo y final del conjunto de muestras de vídeo activas es indicado mediante las secuencias de bits *Start of Active Vídeo (SAV)* y *End of Active Vídeo (EAV).* Se trata de dos códigos utilizados como marcas de sincronización del vídeo, que son detectados mediante una cabecera compuesta por los valores en hexadecimal FF 00 00. Por lo tanto, los píxeles de la imagen no pueden tomar los valores FF o 00 ya que podrían provocar detecciones erróneas de marcas de sincronización.

A continuación, aparece un codeword denominado XYZ que indica el comienzo o final de una línea de vídeo activa (sincronismo horizontal, H), el comienzo o final de un campo o plano de vídeo activo (sincronismo vertical, V), y en caso de que se trabaje con un formato entrelazado, si el campo en cuestión es top o bottom (flag F). Las líneas de vídeo no activo conforman el intervalo de tiempo denominando *vertical blanking*. Las muestras de vídeo no activo, conforman el intervalo de tiempo denominando *horizontal blanking*. Exactamente, el codeword XYZ está definido de la siguiente forma (empleando codewords de 10bits):

- Bit 9 =0,
- bit 8 = F,
- bit 7 = V,
- bit 6 = H,
- resto de bits son bits de protección empleando un código de Hamming:
    - bit 5 = V xor H,
    - bit 4 = F xor H,
    - bit 3 = F xor V,
    - bit 2 = F xor V xor H.

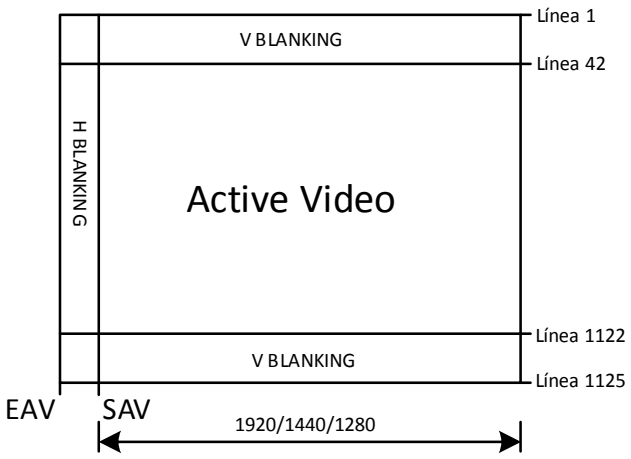


Figura 6.9. Relación entre vídeo activo/blanking para la resolución 1080p según SMPTE 274M.

6.2.2.4 Infraestructura y evolución

La mayoría de dispositivos profesionales utilizados dentro del mercado broadcast utilizan el interfaz SDI para el transporte de la señal de audio y vídeo. La señal SDI es transportada mediante cable coaxial utilizando conectores BNC. Dentro de estos dispositivos se pueden encontrar cámaras, monitores, mesas de mezcla de vídeo, etc. SDI es un interfaz muy difundido actualmente, aunque en un futuro no muy lejano podría ser sustituido por la transmisión de vídeo y audio sin comprimir en tiempo real a través de una red IP utilizando el recientemente publicado (septiembre de 2018) set de estándares SMPTE ST 2110.

SMPTE ST 2110 ofrece una serie de ventajas indiscutibles, como puede ser la eliminación de la necesidad del uso de cable coaxial y reducción del cableado utilizado, pero requiere la renovación de las infraestructuras dentro del centro donde se desee su implantación. Este despliegue conlleva una importante inversión, lo que conlleva que SDI seguirá siendo utilizado durante los próximos años. El uso de vídeo sin comprimir sobre una red IP requiere el uso de una red de fibra óptica con un ancho de banda considerable (10 Gb – 400 Gb), así como el uso de una topología de red adecuada y de dispositivos compatibles con SMPTE ST 2110. Existen ciertos parámetros utilizados en los estudios de televisión que hacen que la transmisión de vídeo y audio sin comprimir no sea una simple comunicación basada en el protocolo IP. Normalmente, debe existir una fuente de reloj común a todo el estudio que sirve de referencia a todos los equipos de monitorización. Dicha referencia es conocida como señal de sincronismo de cuadro o *genlock*. Esta señal es introducida en los diferentes equipos como una entrada que proporciona un sincronismo externo, de forma que el equipo en cuestión proporcione la señal de salida sincronizada con dicha referencia. De esta forma, todas las señales del estudio estarán sincronizadas facilitando las labores de producción de contenidos. Utilizando sistemas basados en cable coaxial, la señal se genera mediante un generador y es distribuida mediante cableado a todos los equipos. En el caso de la transmisión sobre IP, se requiere el uso de un protocolo de comunicación que permita medir los retardos producidos en la comunicación para poder compensarlos. Dicho protocolo es definido por el estándar SMPTE 2059. Es en este punto donde reside principalmente la importancia de utilizar dentro de la red IP dispositivos compatibles con SMPTE ST 2110/ SMPTE 2059, tales como los switches.

6.3 Sistema de captura SDI/PCIe

6.3.1 Arquitectura

En la Figura 6.10 se define la arquitectura del sistema implementado. Como se puede apreciar, la señal de entrada SDI es procesada por la FPGA para extraer el vídeo y el audio presente en la señal y enviarlo a través del bus PCIe al microprocesador. En el microprocesador, la GPU embebida que posee, utiliza el vídeo recibido a través del bus PCIe para realizar un pre-análisis de la imagen y aplicar los algoritmos de clasificación comentados en apartados anteriores. La GPU también es utilizada para realizar la conversión entre el espacio de color 4:2:2 *UYVY interleaved* del vídeo que procede de la señal SDI al espacio de color 4:2:0 *YUV planar*, que es el que requiere la codificación y los algoritmos de análisis de la imagen. Dicha imagen puede verse sometida a un filtrado, si el algoritmo de eliminación de información no perceptible comentado en el apartado 5.3.1.6.2 fue aplicado, o a un cambio de resolución mediante procesos de escalado. Por lo tanto, la GPU proporciona al procesador a cargo de la codificación, la imagen de entrada 4:2:0 YUV planar (pudiendo haber sido escalada o filtrada) y una serie de datos obtenidos del análisis de la imagen para llevar a cabo los algoritmos de optimización comentados en el capítulo 5. En el procesador se lleva a cabo el proceso de codificación HEVC propiamente dicho.

En el diseño implementado en la FPGA se ha eliminado la necesidad de utilizar una memoria externa para este dispositivo, de forma que las imágenes extraídas de la señal SDI son directamente almacenadas en la memoria del microprocesador, que es accesible por la FPGA desde el bus PCIe. Prescindir de la memoria externa de la FPGA presenta una serie de ventajas, como son la reducción del precio del sistema, reducción del consumo y la reducción del retardo asociado a la entrada de vídeo.

La utilización de la FPGA proporciona baja latencia en el procesado, dispone del hardware que permite conectarse a una señal SDI para la extracción del audio y del vídeo, y permite implementar algoritmos de pre-procesado de la imagen para mejorar la calidad de la señal recibida. Por otro lado, al estar implementados los algoritmos de análisis de la imagen en lenguaje OpenCL, en un futuro, los kernels desarrollados podrían ejecutarse en la FPGA en lugar de en la GPU si se necesitase destinar las unidades lógicas de la GPU a otros procesos.

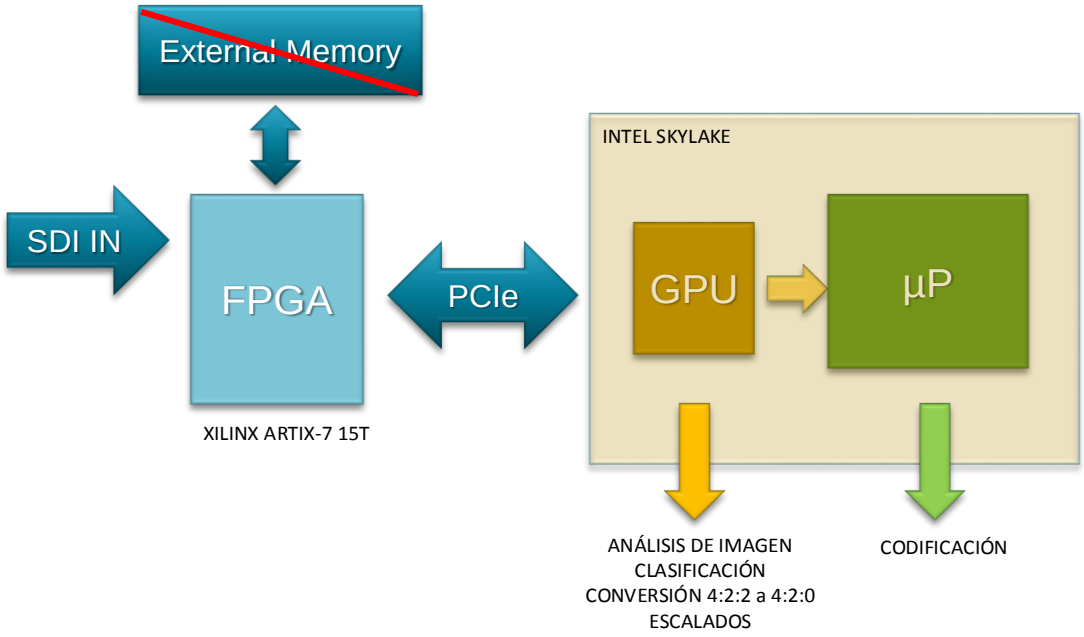


Figura 6.10. Arquitectura del sistema de captura de vídeo y codificación.

6.3.2 Transceptores de señal

A la hora de crear un sistema basado en FPGA que sea capaz de extraer audio y vídeo de un interfaz SDI y comunicarse con un microprocesador a través del bus PCIe, es imprescindible utilizar un modelo de FPGA que incluya los transceptores de señal que se requieren para implementar estas funcionalidades.

Tanto PCIe como SDI, utilizan una comunicación serie manejando tasas de transferencia de datos superiores a 1 Gb/s. Los transceptores de señal son el hardware dentro de la FPGA que permite que un dispositivo pueda alcanzar tasas de transferencia de varios gigabits/segundo en una comunicación. Para tal propósito utilizan una señal diferencial, que es menos susceptible al ruido.

En los dispositivos del fabricante Xilinx, los transceptores de señal son conocidos como *Multi-Gigabit Transceivers (MGT)* o *multi-gigabit Serializer/Deserializers (SERDES)*, permitiendo transmitir en serie datos generados en paralelo desde la FPGA a otros dispositivos y la operación contraria, recibir datos en serie y convertirlos a datos en paralelo para su procesado dentro de la FPGA. Dentro de las familias de bajo coste de Xilinx, la familia Spartan-7 no dispone de MGT, por lo que no se puede utilizar para la aplicación que se pretende desarrollar. Por el contrario, la familia Artix-7 dispone de dispositivos cuyo número de MGT varía entre 2 y 16. MGT es el nombre genérico para los transceptores de señal en Xilinx, pero dentro de cada familia varía el nombre utilizado en función de la tecnología empleada, siendo GTP en Artix-7 y GTX en Kintex-7. En la Tabla 6.4 se muestran los dispositivos que proporciona Xilinx para la familia Artix-7.

Tabla 6.4. Características asociadas a la familia Artix-7 de Xilinx. Extraída de [5].

		Transceiver Optimization at the Lowest Cost and Highest DSP Bandwidth (1.0V, 0.95V, 0.9V)							
	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280	52,160	75,520	101,440	215,360
	Slices	2,000	2,600	3,650	5,200	8,150	11,800	15,850	33,650
	CLB Flip-Flops	16,000	20,800	29,200	41,600	65,200	94,400	126,800	269,200
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400	600	892	1,188	2,888
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50	75	105	135	365
	Total Block RAM (Kb)	720	900	1,620	1,800	2,700	3,780	4,860	13,140
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5	5	6	6	10
I/O Resources	Maximum Single-Ended I/O	150	250	150	250	250	300	300	500
	Maximum Differential I/O Pairs	72	120	72	120	120	144	144	240
Embedded Hard IP Resources	DSP Slices	40	45	80	90	120	180	240	740
	PCIe® Gen2[1]	1	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1
	GTP Transceivers (6.6 Gb/s Max Rate)[2]	2	4	4	4	4	8	8	16
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L

Si se limita la recepción de vídeo SDI a una máxima resolución de 1080p60, que implica un movimiento de datos próximo a 3 Gb/s, sólo se requiere el uso de un GTP, ya que éste es capaz de alcanzar una tasa de hasta 6.6 Gb/s.

Por otro lado, una vez extraído el vídeo válido (sin blanking) del interfaz SDI, para transferirlo al procesador encargado de la codificación con una precisión de 8 bits por píxel y utilizando muestreo de color 4:2:2; la tasa de datos requerida se reduce a 1920 x 1080 x 8 x 60 x 2 = 1990656 Kbits/segundo (< 2 Gb/s). Por lo tanto, para el interfaz PCIe se requiere únicamente un GTP. Como se puede observar en la Tabla 6.4, la familia Artix-7 además incorpora un módulo hardware PCIe Gen2, lo que simplificará enormemente la utilización de este bus al encargarse de la gestión de las capas Physical (incluido el interfaz eléctrico interno a la FPGA) y Data Link. Al ser Gen2, con un único lane se alcanzará una tasa de transferencia de datos de hasta 4 Gb/s, lo que supera el ancho de banda requerido por la aplicación. Si fuese necesario emplear más lanes para aumentar la tasa de transferencia de datos, hay que considerar que es necesario emplear un GTP extra para cada lane.

Por lo tanto, la aplicación que hace uso del interfaz SDI y del bus PCIe requerirá al menos dos GTP. En la Tabla 6.4, el modelo XC7A12T dispone de 2 GTP y es el más económico de la familia Artix-7, pero también dispone de una lógica muy reducida (12800 *logic cells*), por lo que se decidió utilizar el modelo XC7A15T.

En [6] se puede encontrar una descripción detalla de los GTP presentes en la familia Artix-7 de Xilinx. Dentro de los aspectos que se abordan en este documento, se hará hincapié en la distribución de relojes necesaria para un correcto funcionamiento de los GTP. En la Figura 6.11, se muestra un GTP Quad compuesto por cuatro GTP (GTPE2_CHANNEL) y una instancia GTPE2_COMMON, que es la encargada de suministrar el reloj a los cuatro GTP.

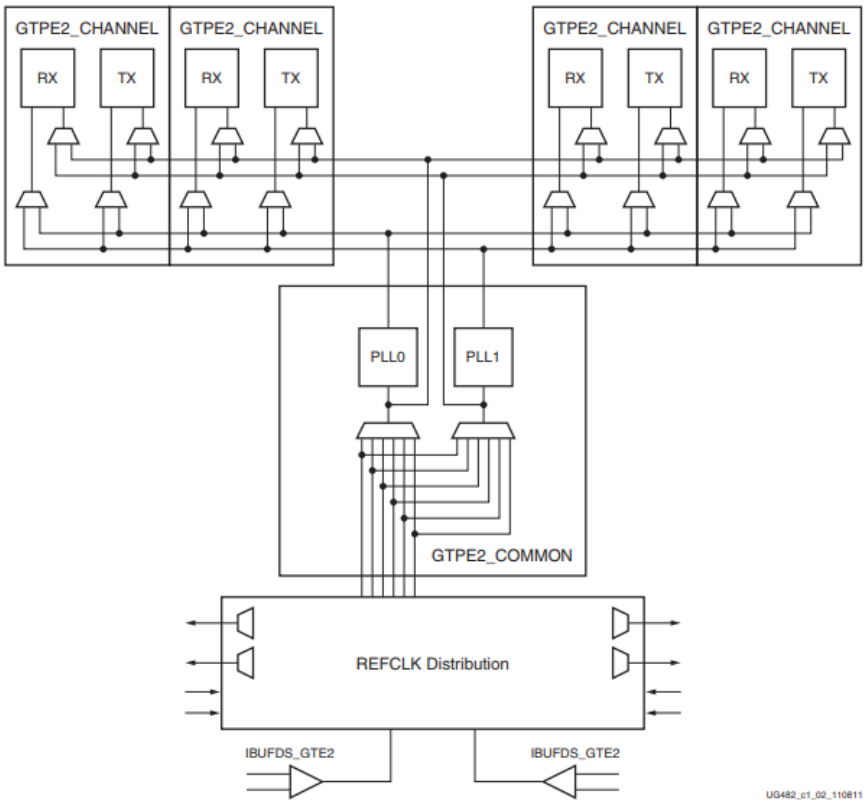


Figura 6.11. Configuración Quad de los GTP y la distribución de relojes existente. Extraído de [6].

Los dos componentes IBUFDS_GTE2, deben conectarse cada uno a una pareja de pines de reloj diferencial. Uno proporcionará la referencia de reloj para el PLL1 que a su vez proporcionará el reloj al GTP de la recepción SDI, y el otro proporcionará otra referencia distinta para el PLL0 que proporciona reloj al GTP utilizado para la comunicación PCIe. Es necesario el uso de dos referencias de reloj distintas, debido a que la frecuencia de cada reloj es diferente. Para los módulos SDI se requiere una frecuencia de 148.5 MHz para poder recibir 3G-SDI, mientras que a los módulos PCIe hay que proporcionar el reloj de sistema PCIe (proporcionado por el microprocesador) que en nuestro caso es de 100 MHz. Para que PLL0 y PLL1 reciban la frecuencia de referencia adecuada, se deben configurar correctamente los multiplexores de reloj que se muestran en la Figura 6.12. Suponiendo que los pines MGTREFCLK0P/N están conectados al reloj de 100 MHz y los pines MGTREFCLK1P/N al de 148.5 MHz, el multiplexor conectado al PLL0 debería ser configurado con un "1" y el multiplexor a la entrada del PLL1 con un "2". Una vez configurado el reloj que se proporciona como referencia a PLL0 y PLL1, es necesario configurar qué reloj se proporcionará como referencia a cada GTPE2_CHANNEL. Para ello hay que configurar los multiplexores que aparecen en la Figura 6.13, mediante las señales RXSYSCLK_SEL y TXSYSCLK_SEL.

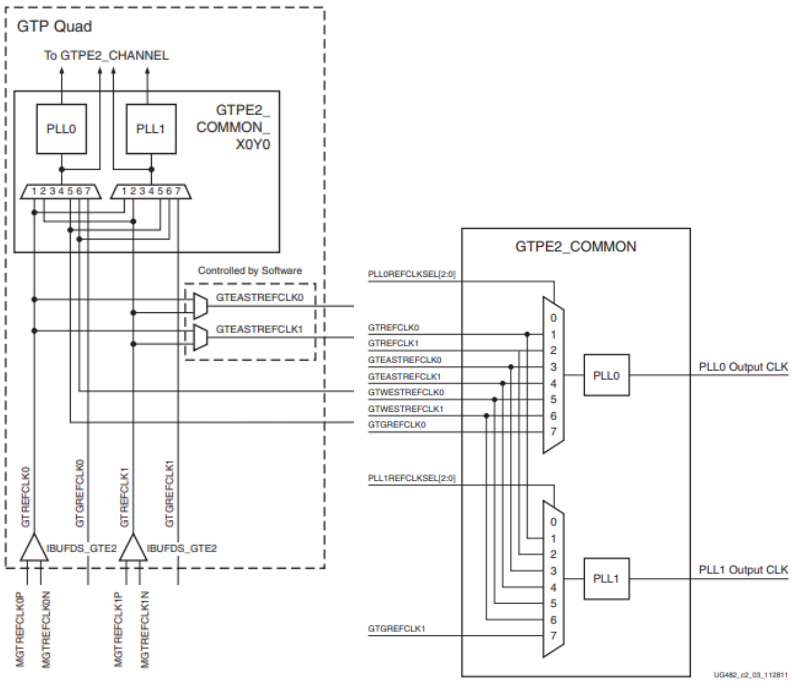


Figura 6.12. Distribución de relojes de referencia para la instancia GTPE2_COMMON.
Extraído de [6].

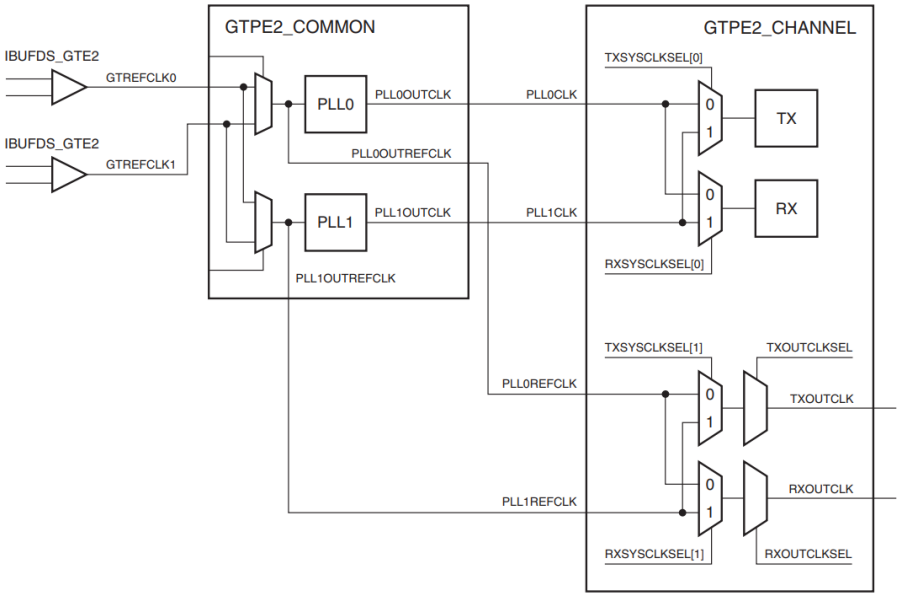


Figura 6.13. Distribución de relojes de referencia para la instancia GTPE2_CHANNEL.
Extraído de [6].

Como ya se ha indicado, PLL0 se utilizará para PCIe y PLL1 para la recepción SDI. Por lo tanto, para el GTP que se utilice en la implementación PCIe hay que seleccionar tanto para transmisión como para recepción, el reloj que proviene del PLL0 (TXSYSCLKSEL[0] y RXSYSCLKSEL[0] = 0). El GTP dedicado al SDI sólo hará uso

de la parte de recepción, por lo que basta con configurar el multiplexor de la parte de recepción (RXSYSCLKSEL[0] = 1).

6.3.3 Recepción SDI

Para la implementación de la parte de recepción SDI, resulta fundamental la lectura previa de la nota de aplicación de Xilinx sobre la implementación de interfaces SMPTE SD-SDI, HD-SDI y 3G-SDI en la familia Artix-7 [7]. Se trata de un documento que explica cómo realizar la implementación tanto para la recepción como para la parte de transmisión SDI. La aplicación desarrollada únicamente hará uso del interfaz de recepción, que será el que se utilice para extraer el vídeo y audio de la señal SDI.

Xilinx proporciona un core SDI denominado *SMPTE SD/HD/3G-SDI LogiCORE IP* y un asistente (*wizard*) para instanciar un GTP conforme a una determinada funcionalidad, pero ambos deben de ser complementados con lógica adicional para su correcto funcionamiento. En la nota de aplicación mostrada en [7], se explican los módulos necesarios y su funcionalidad, además se indica un enlace [8] donde se puede descargar el código necesario para verificar el diseño de referencia y realizar las modificaciones que se estimen oportunas. En la Figura 6.14 se muestra el esquema general de un interfaz RX/TX SDI.

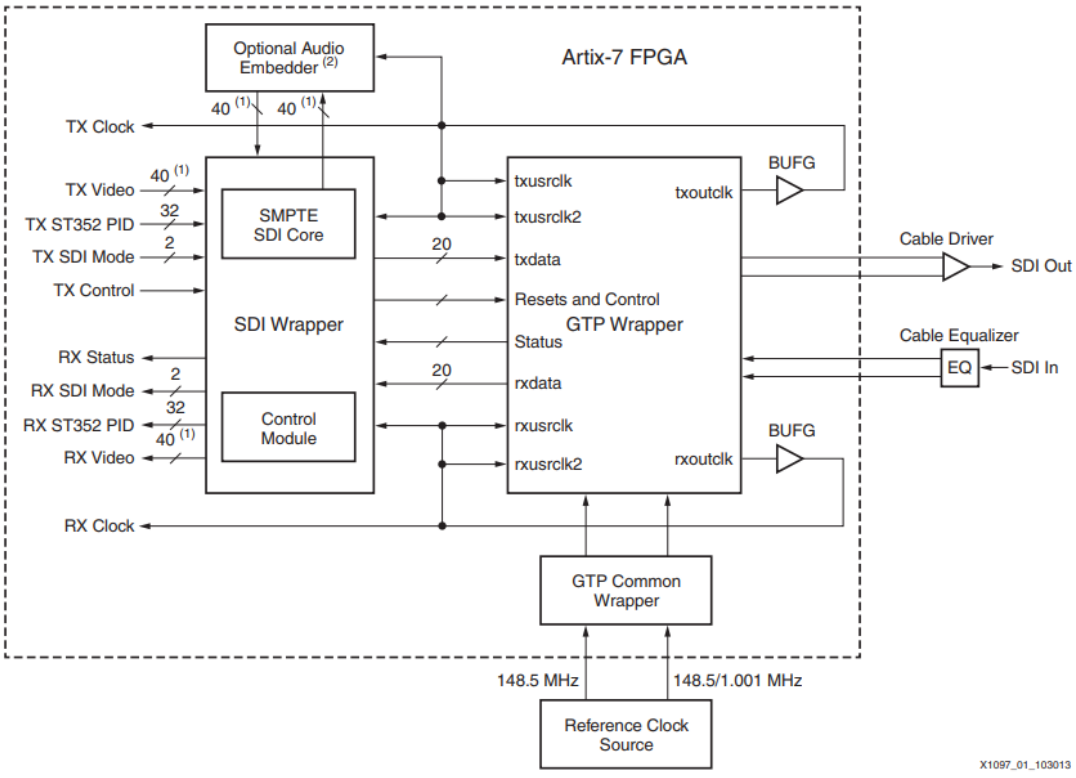


Figura 6.14. Diagrama de bloques de un interfaz SDI.
Extraído de [7].

Para el caso de la recepción SDI, se requiere un ecualizador de cable, siendo éste un componente externo a la FPGA. En el caso del diseño de referencia [7-8] se usa el componente LMH0387 de Texas Instruments [9]. Este componente es ampliamente

utilizado en el mercado broadcast para realizar la ecualización necesaria en los datos recibidos a través del cable coaxial en la captura SDI.

Dentro de las señales de salida que proporciona el módulo SDI wrapper se encuentra la señal RX SDI Mode, que permitirá saber qué formato de vídeo se está recibiendo. La señal RX Video proporciona la señal de vídeo extraída de la señal SDI. En la parte correspondiente al blanking de esta señal de vídeo puede haber datos auxiliares que incluyan audio. Si se desea extraer el audio, se requiere del uso de un demultiplexor de audio y vídeo. En [10] se explica el proceso de demultiplexación y se proporciona un enlace a un ejemplo de referencia para una FPGA Virtex-5 de Xilinx. Al tratarse de código VHDL y no hacer uso de instancias específicas de la familia Virtex-5, puede utilizarse sin problemas en una Artix-7. La señal RX Status está compuesta por varios flags que indican situaciones de error CRC y otros posibles problemas encontrados en la recepción SDI. El módulo SDI wrapper extrae paquetes según la norma SMPTE 352 [11], proporcionándolos por medio de la señal de salida RX ST352 PID. Dentro de la información que proporcionan estos paquetes, destacan el frame rate del formato de vídeo, la norma SMPTE a la que pertenece el formato de vídeo recibido (SMPTE 274M, SMPTE 259M, SMPTE 296M, SMPTE 425M-A, etc.), la relación de aspecto (4:3 o 16:9) y el muestreo de color utilizado. El reloj RX Clock debe de ser utilizado como reloj de usuario para utilizar las salidas del módulo de recepción SDI. RX Clock es igual al reloj proporcionado por el módulo GTP wrapper, que se corresponde con el reloj extraído de la señal SDI tras atravesar un componente BUFG, cuya misión es asegurar la correcta distribución de un reloj a través los árboles de distribución de relojes existentes dentro de la FPGA. Al utilizar un BUFG, se reduce el *skew* del reloj, es decir se asegura que el reloj llega a todos los componentes síncronos de la FPGA que lo utilicen con el mismo retardo.

6.3.3.1 Relojes de referencia

La referencia de 148.5/1.001 MHz (≈148.35 MHz) que aparece en la Figura 6.14 es necesaria únicamente para la transmisión en el caso de que se utilicen formatos con frame rates iguales a 29.97 fps o 59.94 fps. Dichos frame rates se obtienen dividiendo 30 fps y 60 fps entre 1.001. Esto se debe a que la transmisión requiere que se le proporcione un reloj exacto al necesario para el formato de vídeo SDI que se vaya a transmitir, conforme a la frecuencia *fs* que se muestra en la Tabla 6.3. Los relojes de 74.25 y 74.18 MHz se obtienen internamente en la FPGA dividiendo entre dos las referencias de 148.5 y 148.35 MHz, respectivamente.

El GTP encargado de la recepción no necesita que se le proporcione una relación de reloj exacta respecto al formato de vídeo que se esté recibiendo en la señal SDI. Esto se debe a que el módulo encargado de la recuperación del reloj y de los datos de la señal SDI puede recibir bitrates con una variación de hasta ±1250 ppm respecto al bitrate fijado por la frecuencia del reloj de referencia. La diferencia entre 148.5 y 148.35 MHz es de 1000 ppm, valor inferior a la tolerancia de la recepción. Por lo tanto, la referencia de reloj del GTP receptor puede ser únicamente 148.5 MHz, y puede proporcionarse mediante un oscilador o un PLL externo a la FPGA. Si se usa un oscilador puede asegurarse que el reloj de referencia del GTP de recepción es siempre estable, eliminando la necesidad de su control para la secuencia de reset en la inicialización del GTP. Por el contrario, si se usa un PLL hay que asegurarse que el GTP se encuentre en reset hasta que el reloj que proporcione el PLL esté estable, lo que puede controlarse conectando la señal rx_refclk_stable a la señal de LOCK del PLL.

La parte receptora del módulo GTP Wrapper genera un reloj recuperado (rxoutclk en la Figura 6.14) que está enganchado a la frecuencia del bitrate SDI recibido. Este reloj es:

- 148.5 MHz para 1080p50/60 y SD PAL/NTSC.
- 148.35 MHz para 1080p59.
- 74.25 MHz para los formatos HD 720p50, 720p25, 1080p25 y 1080i50.
- 74.18 MHz para los formatos HD 720p59, 720p29, 1080p29 y 1080i59.

Para los formatos SD, la frecuencia requerida es de 27 MHz, pero el reloj recuperado que proporciona el módulo es 148.5 MHz. Para resolver esta cuestión, el módulo receptor SDI proporciona una señal de habilitación de reloj con una cadencia 5/6/5/6.

Los GTP requieren de un reloj cuya frecuencia permanezca fija y constante aunque cambie el formato de vídeo SDI con el que se esté trabajando. Este reloj es utilizado para el puerto de reconfiguración dinámica (*Dynamic Reconfiguration Port, DRP*) para medir los retardos de ciertas secuencias internas. Xilinx recomienda que la frecuencia de este reloj sea de al menos 10 MHz. Este reloj puede ser utilizado para todos los interfaces SDI de la FPGA, puesto que no debe guardar ninguna relación con el bitrate con el que se trabaje, pero nunca debe detenerse mientras esté funcionando la aplicación SDI. La frecuencia que finalmente se utilice se debe de indicar en el módulo GTP wrapper.

6.3.3.2 Consideraciones para la recepción de SD-SDI

El bitrate utilizado en SD-SDI es de 270 Mb/s, el cual se encuentra por debajo de la mínima tasa soportada por un GTP. Para poder recibir 270 Mb/s en el GTP receptor se realiza un sobre muestreo aplicado a la señal de entrada a 2.97 Gb/s (11 veces 270 Mb/s). Una unidad de recuperación de datos (*Data Recovery Unit, DRU*), implementada en la lógica programable de la FPGA se encarga de examinar los datos muestreados por el GTP receptor, determina la mejor muestra a utilizar para cada bit y proporciona a su salida el dato recuperado. DRU no es parte del core SDI, pero se proporciona como parte de la aplicación SDI detallada en [7]. Se trata de una versión optimizada del DRU descrito en [12], modificado para manejar exclusivamente el sobre muestreo 11x, lo que reduce la cantidad de recursos de la FPGA que se utilizan.

6.3.3.3 Consideraciones para la detección de los formatos de entrada

El core SDI tiene la capacidad de detectar automáticamente el modo SDI (SD, HD y 3G) de la señal SDI de entrada. Cuando no es capaz de detectar ningún modo válido, el core SDI comienza una secuencia recorriendo los tres posibles modos SDI hasta que detecta datos válidos en la salida del GTP a cargo de la recepción. Pero una vez que se ha detectado un modo HD-SDI o 3G-SDI, el core SDI no es capaz de determinar si el bitrate de la señal de entrada es 1.485 Gb/s o 1.485/1.001 Gb/s en HD ni 2.97 Gb/s o 2.97/1.001 Gb/s para 3G. Para solucionar este problema la aplicación SDI [7-8] incorpora un detector de bitrate incluido en el módulo SDI wrapper que puede distinguir entre las diferentes opciones. Para que la detección del bitrate funcione correctamente, al módulo SDI wrapper se le debe proporcionar un reloj de frecuencia fija. La frecuencia de dicho reloj debe de ser superior a 10 MHz e inferior a 150 MHz, la cota superior está impuesta porque frecuencias superiores a 150 MHz provocarán que el diseño tenga dificultad en cumplir las restricciones relativas a la máxima frecuencia de operación.

6.3.4 Comunicación PCIe

6.3.4.1 Fase de enumeración

La enumeración PCIe es el proceso mediante el cual el Root Complex detecta todos los dispositivos conectados al bus PCIe. Una vez que el dispositivo es reconocido, se accede a la información relativa al tamaño del BAR y se reserva en la memoria de sistema el espacio requerido. Posteriormente, se establece una fase de configuración en la que se habilita al dispositivo como Bus Máster.

Tras la enumeración, se establece una etapa de negociación entre el Root Complex y el endpoint para determinar la compatibilidad entre las distintas capacidades de los dos dispositivos y establecer la configuración que mayores prestaciones ofrezca.

En la Figura 6.15 se muestra una captura de pantalla obtenida tras ejecutar en un sistema operativo basado en Linux el comando *lspci –vv* una vez finalizadas las etapas de enumeración, negociación y configuración. Por medio de este comando se obtiene un listado de los dispositivos PCI existentes, y el parámetro –vv permite obtener un listado exhaustivo de sus características más relevantes.

```
01:00.0 Memory controller: Xilinx Corporation Device 7025
        Subsystem: Xilinx Corporation Device 0007
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin A routed to IRQ 128
        Region 0: Memory at df000000 (32-bit, non-prefetchable) [size=4K]
        Capabilities: [40] Power Management version 3
                Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0+,D1+,D2+,D3hot+,D3cold-)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [48] MSI: Enable+ Count=16/16 Maskable- 64bit+
                Address: 00000000fee00418  Data: 0000
        Capabilities: [60] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 512 bytes, PhantFunc 0, Latency L0s <64ns, L1 unlimited
                        ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FLReset- SlotPowerLimit 25.000W
                DevCtl: Report errors: Correctable- Non-Fatal- Fatal- Unsupported-
                        RlxdOrd- ExtTag- PhantFunc- AuxPwr- NoSnoop+
                        MaxPayload 256 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr- TransPend-
                LnkCap: Port #0, Speed 5GT/s, Width x1, ASPM L0s, Exit Latency L0s unlimited, L1 unlimited
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp-
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk+
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 5GT/s, Width x1, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                DevCap2: Completion Timeout: Range B, TimeoutDis-, LTR-, OBFF Not Supported
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled
                LnkCtl2: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-
                         Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                         Compliance De-emphasis: -6dB
                LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete-, EqualizationPhase1-
                         EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
        Capabilities: [100 v1] Device Serial Number 00-00-00-01-01-00-0a-35
        Kernel driver in use: pavc
        Kernel modules: pavc
```

Figura 6.15. Resultado de la consola tras aplicar el comando lspci –vv.

Cabe destacar dentro de la información mostrada en la Figura 6.15, la descripción inequívoca del dispositivo como *Memory Controller: Xilinx Corporation Device 7025*, información que ha sido derivada directamente de los registros del espacio de memoria de configuración: Class Code, Vendor ID y Device ID. A continuación, se muestra el campo Subsystem Vendor ID. Se puede observar también que se trata de un dispositivo configurado como BusMaster (*Control: I/O+ Mem+ BusMaster+*), el tamaño del BAR asignado de 4 KB y su mapeo en memoria (Region 0: Memory at df000000 (32-bit, non-prefetchable) [size=4K]).

Dentro de las capacidades del dispositivo se muestra que es un dispositivo PCIe v2 Endpoint, que soporta hasta 16 vectores de interrupción MSI y Max_Payload_Size = 512 bytes. Aunque este último parámetro sea de 512 bytes (indicado en *DevCap*), tras negociar con el microprocesador se ha fijado a 256 bytes (*DevCtl*) al no soportar éste un Max_Payload_Size de 512 bytes. Otras características interesantes son las relacionadas con las capacidades del link, donde se indica que se soporta una velocidad de 5GT/s y que se ha implementado un único lane (*x1*). En los apartados *LnkSta y LnkCTl* se muestran las capacidades finalmente negociadas y que se utilizarán en la comunicación FPGA-microprocesador.

6.3.4.2 Implementación

Para transmitir o recibir TLP es necesario utilizar el interfaz de la capa de transacción (*Transaction Layer Interface*) que proporciona el core PCIe de Xilinx. Básicamente, se requiere que los paquetes TLP sean conformados según el formato que establece la especificación PCIe [1], y comunicarse con el core PCIe conforme al protocolo que requiere [13]. En [13] se realiza una descripción detallada del core PCIe, cuya lectura resulta indispensable para establecer una comunicación a través del bus PCIe. Una vez generado el core PCIe mediante el wizard correspondiente de Xilinx, se puede utilizar un proyecto de ejemplo que servirá como punto de partida. Dicho proyecto es generado automáticamente. Se trata de un ejemplo que usa transacciones tipo PIO, por lo que no se conseguirá un elevado ancho de banda en la comunicación, pero permite comprobar que el hardware funciona correctamente y entender el funcionamiento de los distintos componentes involucrados en la comunicación microprocesador-FPGA.

Dentro de las opciones disponibles para la generación del core, es seleccionable el número de lanes a utilizar, la máxima velocidad del link en giga-transferencias por segundo (para el caso de 1 lane, 2.5 GT/s o 5.0 GT/s), la frecuencia del reloj de referencia PCIe, los identificadores del diseño (Vendor ID, Device ID, Revision ID, Subsystem Vendor ID, Sybsystem ID, Class Code, etc.), número de BAR y tamaño de cada BAR en Kilobytes, Max_Payload_Size, etc. También se puede habilitar el uso de interrupciones tipo MSI y el número de vectores de interrupción asociado.

El proyecto de ejemplo incluye un fichero de restricciones de posicionamiento en el que se indica el pinout asociado a las diferentes lanes. Dicho posicionamiento debe ser modificado conforme al pinout de la tarjeta que se esté utilizando.

En [14] se presenta una nota de aplicación para facilitar la implementación de diseños PCIe que implementen un sistema BMD. Dicha aplicación ha sido utilizada como base del diseño realizado. En la Figura 6.16 se muestra el diagrama de bloques del diseño realizado para implementar un sistema BMD.

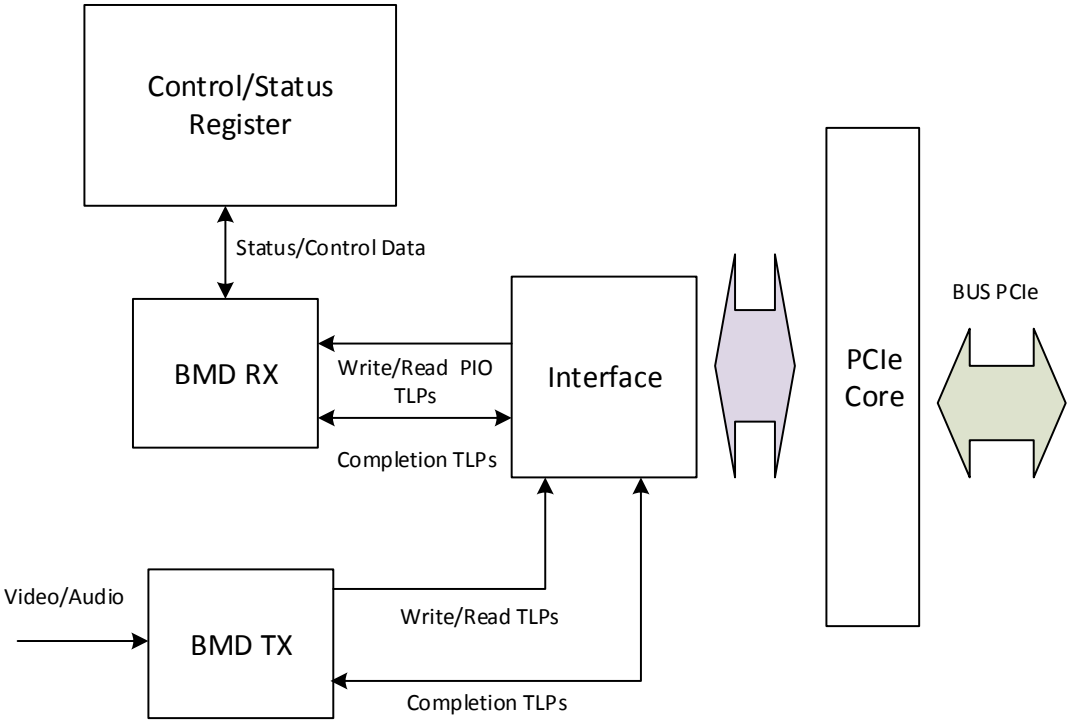


Figura 6.16. Diagrama de bloques del diseño realizado.

El módulo PCIe Core es el core que proporciona Xilinx para instanciar el módulo hardware que se encarga de implementar esta función. El módulo BMD RX es el encargado de recibir los TLP que provienen de otros dispositivos PCIe y traducir las peticiones recibidas a comandos internos de la FPGA. BMD RX captura e interpreta accesos de tipo PIO (de un único DW) tanto de lectura como de escritura. Se trata de TLP que son enviados al endpoint con el propósito de monitorizar su estado y realizar tareas de control/configuración por medio de los registros asignados para tal propósito (Control/Status Registers). El módulo BMD RX también genera los TLP de tipo completion que incluyen el dato leído de los registros. El módulo BMD TX es el encargado de generar los TLP que se transmiten al microprocesador desde la FPGA, siendo éstos peticiones de escritura o de lectura a la memoria del sistema y los datos que se desean almacenar en la memoria del sistema. En la aplicación desarrollada, dichos datos se corresponden con el vídeo y audio extraídos de la señal SDI que serán transmitidos al microprocesador para realizar el análisis de la imagen y la codificación.

6.3.5 Intercambio de datos SDI/PCIe

6.3.5.1 Escritura/lectura de las memorias de vídeo

En la Figura 6.17, se muestra el diagrama de bloques del diseño que permite transferir el vídeo y audio recibidos por el interfaz SDI al microprocesador a través del bus PCIe.

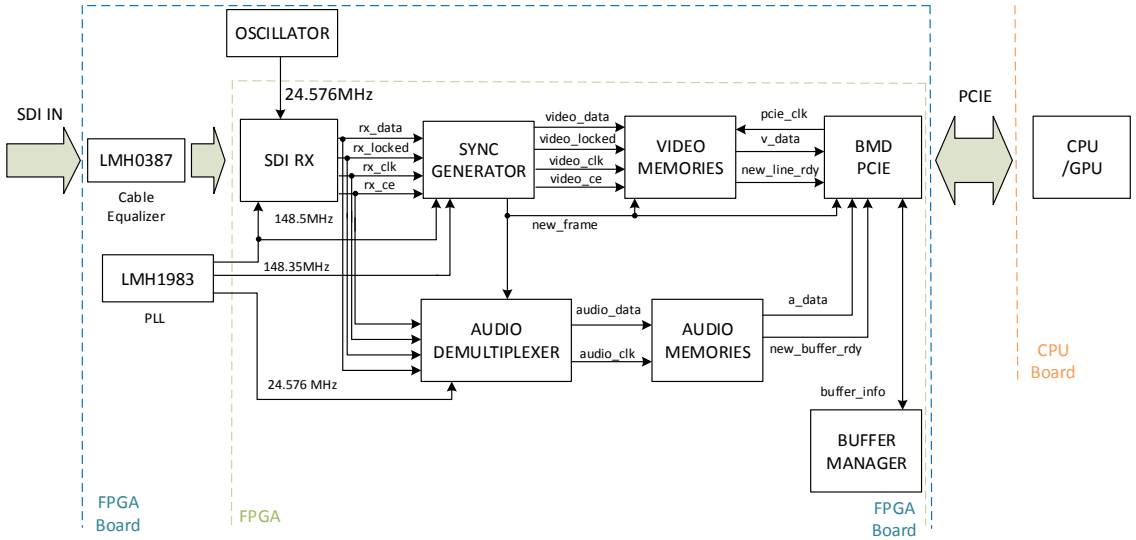


Figura 6.17. Diagrama de bloques de la aplicación de captura SDI/transmisión PCIe.

El vídeo válido procedente del módulo de recepción SDI es almacenado en las memorias (*VIDEO MEMORIES*) a la frecuencia de reloj que impone *rx_clk*, siendo ésta la impuesta por el bitrate del formato de vídeo que se esté recibiendo. A su vez, dichos datos serán leídos para su transmisión a través del bus PCIe a la frecuencia que impone *pcie_clk*. Los DW que se transmiten son de 32 bits. Para que cada acceso a las memorias genere un DW, los datos contenidos por las memorias serán de 32 bits. Por lo tanto, si los datos almacenados en las memorias son de 32 bits y los píxeles que componen la imagen son de 8 bits, cada posición de memoria contendrá 4 píxeles.

En formatos HD el módulo de recepción SDI proporciona en cada ciclo de reloj dos muestras de vídeo, una se corresponde a la luminancia (Y) y otra a la crominancia (Cb o Cr, alternativamente), mientras que en formatos SD en cada ciclo de reloj se proporciona una única muestra. Por lo tanto, en SD se irán alternando muestras de crominancia y de luminancia por el mismo bus. Los datos proporcionados por el módulo de recepción SDI son almacenados en registros intermedios hasta conformar los 4 bytes que componen el dato a almacenar en las memorias. Una vez generado el dato de 32 bits, éste se lee de los registros y se almacena en memoria.

Las transferencias de datos de vídeo desde la FPGA al microprocesador a través del bus PCIe, se realizan mediante diversos payloads hasta completar el tamaño de una línea. Por lo tanto, para que la FPGA inicie la transferencia de una línea, se espera a que haya finalizado el almacenamiento de una línea de vídeo completa (luminancia y crominancia) en memoria. Siendo el muestreo de color 4:2:2, una línea completa de vídeo constará de un número de muestras de luminancia igual al número de píxeles que componen el ancho de la imagen, y el mismo número de muestras de crominancia (Cb y Cr). Por lo tanto, el número de muestras de luminancia y de crominancia que componen una línea será igual al doble del ancho de la imagen. Por ejemplo, para la resolución 1080p, el número de muestras de una línea es 1920x2 = 3840 bytes, que se corresponden con 960 datos de 32 bits.

El módulo BMD TX recibe una señal del módulo de recepción SDI que indica el comienzo de un nuevo plano (*new_frame*). Dicha señal se genera detectando un comando SAV referente al comienzo de la primera línea de vídeo activo, y se utiliza como señal de reset a nivel de plano de todos los módulos. Entre otros usos, el reset permite inicializar a cero los contadores que llevan la cuenta del número de muestras y líneas transmitidas a través del bus PCIe. BMD TX recibe otra señal procedente de las memorias de vídeo que

indica que ya se ha almacenado una línea de vídeo completa (*new_line_rdy*), y que está lista para su transmisión. Dicha señal se utiliza como reset del contador de muestras de línea, y como señal de habilitación del contador de líneas dentro de un plano. También es utilizada para iniciar la lectura de la línea de vídeo de las memorias y su posterior encapsulado en los TLP que serán transmitidos a través del bus PCIe.

Cuando se finaliza la transmisión de un plano de vídeo, se debe generar una interrupción al microprocesador para indicar que dispone de un nuevo plano para su procesado. La interrupción de vídeo es utilizada en el sistema implementado en el microprocesador como una señal de sincronización, operando como un reloj de vídeo que indica cuando un búfer debe de ser transferido al siguiente módulo del *pipeline* que compone el sistema de codificación. Por ello, es necesario que el tiempo entre interrupciones permanezca invariable, para no introducir *jitter* en el reloj de vídeo del sistema. Para conseguir este objetivo, la interrupción realmente no se genera cuando se envía al microprocesador la última muestra de vídeo del plano, ya que puede existir una variación del instante en el que se genera debido a cómo de congestionado esté el bus PCIe. En su lugar, la interrupción siempre se genera con la detección de un paquete de sincronización SDI de tipo SAV que indica el comienzo de la primera línea de vídeo (START_FRAME en Figura 6.18). Como la escritura de la última muestra de vídeo activo de un plano se realiza en el instante anterior a la llegada del paquete SDI tipo EAV que indica el final del vídeo activo de la última línea del plano (END_FRAME), se dispone del tiempo de duración del blanking vertical para la transmisión de la última línea hasta que se genere la interrupción de final de plano (INT). Las diferentes señales de control que se obtienen a partir de los sincronismos de la señal SDI de entrada son ilustradas en la Figura 6.18.

En la Figura 6.19, se indica un ejemplo de pipeline en el que el vídeo recibido por la FPGA es transferido a la aplicación a través del driver correspondiente. En la GPU se realiza la conversión de tipo de muestreo de color, escalados para cambiar la resolución de la imagen si se requiere y el análisis de la imagen en busca de diferentes texturas y tipos de movimiento para extraer información que servirá para implementar una serie de optimizaciones en el codificador HEVC. El bitstream resultante de la codificación es encapsulado antes de ser transmitido a través de una red.

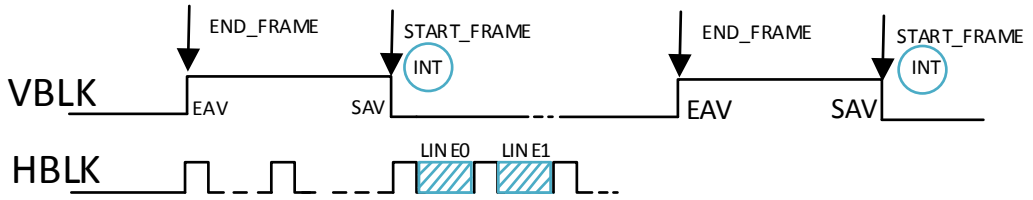


Figura 6.18. Generación de interrupciones a partir de los sincronismos de la señal de entrada.

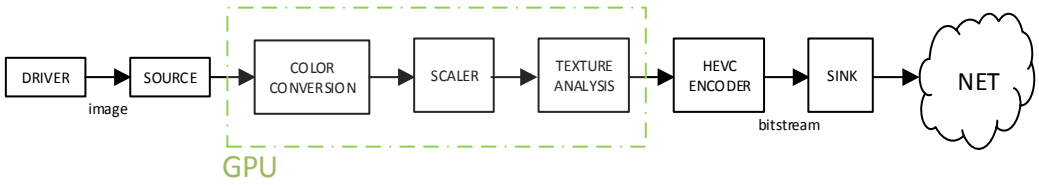


Figura 6.19. Ejemplo de pipeline de aplicación.

6.3.5.2 Protocolo de comunicación con el microprocesador

Desde el punto de vista del microprocesador, la FPGA es un dispositivo PCIe que actúa como controlador de memoria BMD, por lo que requiere de un driver específico para el sistema operativo que se esté utilizando. Dicho driver será el encargado de establecer un protocolo de comunicación con la FPGA para proporcionar el vídeo recibido y la información relativa a la señal SDI de entrada, a la capa de aplicación.

Al haber implementado en la FPGA un diseño BMD, es este dispositivo el encargado de gestionar las transferencias PCIe referentes al almacenamiento del vídeo de entrada en la memoria del sistema. Aun así, el driver debe de gestionar la ventana de memoria a la que tiene acceso la FPGA, realizando la correspondiente reserva y liberación de recursos. Para poder realizar la reserva, la FPGA le informa a través de un registro de estado del formato de vídeo recibido, indicando el ancho y alto de la imagen y el framerate. El espacio de memoria necesario para almacenar un plano de vídeo en bytes es directamente igual a ancho x alto x 2. La multiplicación por 2 se debe a estar utilizando muestreo de color 4:2:2, y no es necesario realizar ninguna adaptación debida al tamaño del píxel al estar trabajando con 8bits para representar cada muestra.

La memoria reservada debe de ser accesible a través del bus PCIe por la FPGA. Para que la FPGA conozca las direcciones de los búferes de memoria a los que tiene acceso, se han implementado unos registros en los que el driver escribe las correspondientes direcciones. La FPGA no debe de escribir en ninguna posición de memoria hasta que el driver le conceda permiso. Esta habilitación se ha implementado a varios niveles. Al trabajar la FPGA como un dispositivo DMA, se ha implementado un registro de habilitación de canal DMA referente a la escritura del vídeo de la entrada SDI. También existe un registro de habilitación de las diferentes interrupciones que puede generar la FPGA. Finalmente, existe una señal que habilita la escritura a cada búfer. Por lo tanto, para cada dirección de memoria que proporciona el driver y que describe un búfer de vídeo, existe una señal de habilitación de escritura desde la FPGA que es controlada por el driver. A su vez, la FPGA indica para cada búfer la finalización de la escritura de un plano de vídeo (búfer completo), de modo que tras la interrupción de fin de escritura de un búfer, el driver pueda saber a qué búfer se asocia la interrupción.

El registro asociado a cada búfer de vídeo para implementar el protocolo de comunicación, se muestra en la Tabla 6.5.

Tabla 6.5. Registro de control asociado a cada búfer.

Field	Bit(s)	Initial Value	R/RW
Enable Write	0	0	RW
Buffer Processed	1	0	RW
Write DMA Address0	31:2	0	RW

A continuación, se resume brevemente el sencillo protocolo de comunicación implementado:

1. El driver habilita el canal DMA de la recepción de vídeo y la interrupción MSI correspondiente, mediante los registros asociados.
2. El driver reserva memoria para tres búferes conforme a la información proporcionada por la FPGA sobre el vídeo de entrada.
3. El driver concede permiso a la FPGA para acceder a dos de los tres búferes (flag *Enable Write = 1* en la Tabla 6.5).
4. La FPGA finaliza la escritura de un búfer, indicando en el campo del registro correspondiente qué búfer ha finalizado (*Buffer Processed = 1*) y generando una interrupción de vídeo. Inmediatamente después, comienza la escritura de un nuevo búfer, por ello requiere tener disponibles siempre dos búferes.
5. Una vez recibida la interrupción, el driver indica a la FPGA que el búfer de vídeo recibido ya no está disponible para su escritura, desactivando el campo del registro correspondiente (*Enable Write = 0*) y borrando el flag de escritura finalizada que activó la FPGA (*Buffer Processed = 0*).
6. El driver proporciona un nuevo búfer a la FPGA para su futura escritura.

En el caso de que la FPGA proporcione audio y vídeo al microprocesador, es necesario el uso de registros que indiquen marcas de tiempo (*timestamps*) que permitan la correcta sincronización de audio y vídeo en la aplicación. Se trata de indicaciones de tiempo que permiten asociar los búferes de audio con determinados búferes de vídeo. El protocolo de comunicación de los búferes de audio es equivalente a lo comentado para el vídeo, con las peculiaridades que se indican posteriormente en el apartado 6.3.6.

En la explicación anterior, se está suponiendo que existe una señal de vídeo válida conectada a la entrada SDI, pero puede existir el caso en el que no haya ninguna señal conectada o que el formato de vídeo de la señal de entrada no sea soportada. Por ello, se implementa un nuevo registro que indica si hay señal de vídeo conectada (que se puede saber gracias a una señal que proporciona el componente externo LMH0387) y si el formato de vídeo es soportado, ya que puede que la señal sea válida pero que el formato de vídeo no sea soportado por el diseño. Un ejemplo de formato de vídeo no soportado es el formato 3G-Level B Dual Stream, que implementa en un canal 3G-SDI dos señales independientes SMPTE 292M (720p de hasta 60fps o 1080i/1080p de hasta 30fps). El módulo de recepción SDI soporta esta funcionalidad, pero no se ha implementado ni el almacenamiento ni la transmisión de este formato en el resto de módulos que componen el diseño realizado. Por otro lado, el módulo de recepción SDI tarda un período de tiempo en engancharse a la señal SDI de entrada y proporcionar datos válidos. Por ello, se indica en otro campo adicional si el módulo SDI está enganchado, lo que indica que la información que se proporciona al driver respecto a la señal de entrada es válida. Cada vez que el módulo SDI se engancha o desengancha, se genera una interrupción MSI para informar al driver. En el Apéndice 1, se indican y describen los diferentes registros que se han implementado en la FPGA.

Debido a la limitación de no realizar peticiones de lectura/escritura PCIe que crucen "4KB boundaries", en los formatos SD (una línea de vídeo son 720x2 = 1440 bytes) la imagen debe de ser almacenada en la memoria del sistema con un salto de línea de 768x2 = 1536 bytes, para que todas las transferencias estén alineadas a 256 bytes y correspondientemente a 4 KB. En la Figura 6.20 se ilustra cómo se almacena una imagen en memoria para formatos SD.

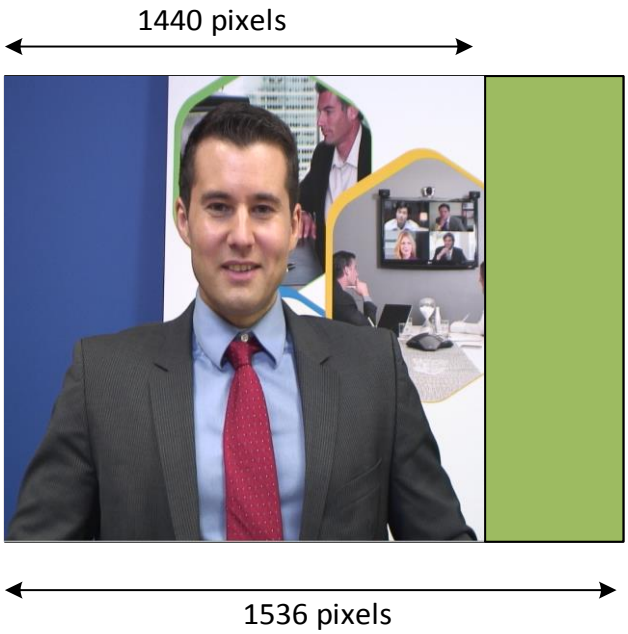


Figura 6.20. Almacenamiento de la imagen en formatos SD.

En la Figura 6.17, el módulo *BUFFER_MANAGER* se encarga de implementar el protocolo de comunicación en la FPGA. Por lo tanto, es el encargado de determinar en qué búfer de la memoria de sistema se deben escribir las muestras de vídeo que se están extrayendo de la entrada SDI y tras finalizar por completo la escritura de un búfer, escribir en el registro correspondiente qué búfer se ha completado e indicar al módulo *BMD PCIE* que genere la interrupción MSI. Para determinar en qué búfer escribir, se consultan de forma secuencial los registros asociados a cada búfer que indican si está habilitada la escritura en dicho búfer por parte del driver. Se trata de un procedimiento de consulta circular, de manera que el orden de consulta a seguir cuando en la FPGA sólo se soportan tres búferes sería 0, 1, 2, 0, 1… Si no hubiese ningún búfer disponible debido a que no está habilitada la escritura en ellos, no se escribiría en la memoria del sistema ningún dato por lo que el plano de vídeo correspondiente sería descartado. Si estando habilitada la escritura de los búferes, éstos ya fueron escritos por la FPGA sin que el host los leyese, se almacenará el plano de vídeo en el siguiente búfer que corresponda rellenar y se indicará al driver un error de tipo *overflow* asociado al búfer utilizado.

6.3.5.2.1 Protocolo de comunicación con el driver en caso de desconexión del cable de entrada y cambio de formato del vídeo

Como se ha indicado previamente, las interrupciones de la FPGA referentes al fin de escritura de un búfer de vídeo son utilizadas por el microprocesador como reloj de vídeo que permite transferir los búferes a través de los diferentes módulos que componen el pipeline del sistema.

La detención de la generación de las interrupciones, provocaría un funcionamiento anómalo en la capa de aplicación, hecho que se ha corroborado empíricamente. Por ello, resulta necesaria la implementación de un mecanismo que permita que la FPGA continúe generando interrupciones aunque la señal de entrada se desconecte. Esto se consigue mediante un proceso implementado en la FPGA que permite replicar los sincronismos que se estaban generando conforme al último formato de vídeo recibido antes de la desconexión del cable. Si el equipo se enciende sin señal conectada, se generan sincronismos conforme a un formato de vídeo por defecto.

Para la generación de dichos sincronismos, se implementa un mecanismo que continuamente va analizando mediante contadores la duración del vídeo válido y del blanking asociado al vídeo de entrada. Dicha información es continuamente actualizada, y en caso de desconexión del cable es utilizada para replicar la señal de entrada y generar interrupciones con la misma cadencia. Las muestras de vídeo válido son sustituidas por una señal de vídeo de color negro. De esta forma, si el equipo está transmitiendo el vídeo a través de una red de comunicaciones y se desconecta el cable o se apaga la cámara, en el equipo receptor simplemente se recibirá una señal de vídeo de color negro, pero no habrá problemas en la transmisión.

La generación de interrupciones se realiza con el reloj RX Clock que proviene del módulo de recepción SDI, pero en el caso de desconexión de la señal de entrada, este reloj deja de proporcionarse. Por ello, la FPGA debe de hacer uso de un PLL externo del que se puedan derivar las frecuencias de reloj necesarias. Estas frecuencias de reloj son 148.5 MHz, 148/1.001 MHz, 74.25 MHz y 74.25/1.001 MHz. El PLL externo proporcionará las dos primeras, y mediante señales de habilitación de reloj se dividirán entre dos para generar las otras dos frecuencias necesarias. De esta forma, en cuanto el módulo SDI se desenganche de la señal de entrada, se conmutará a los relojes del PLL externo para proporcionar la correcta sincronización. En la Figura 6.21 se ilustra la generación de sincronismos para el caso en el que la señal de entrada se desconecta.

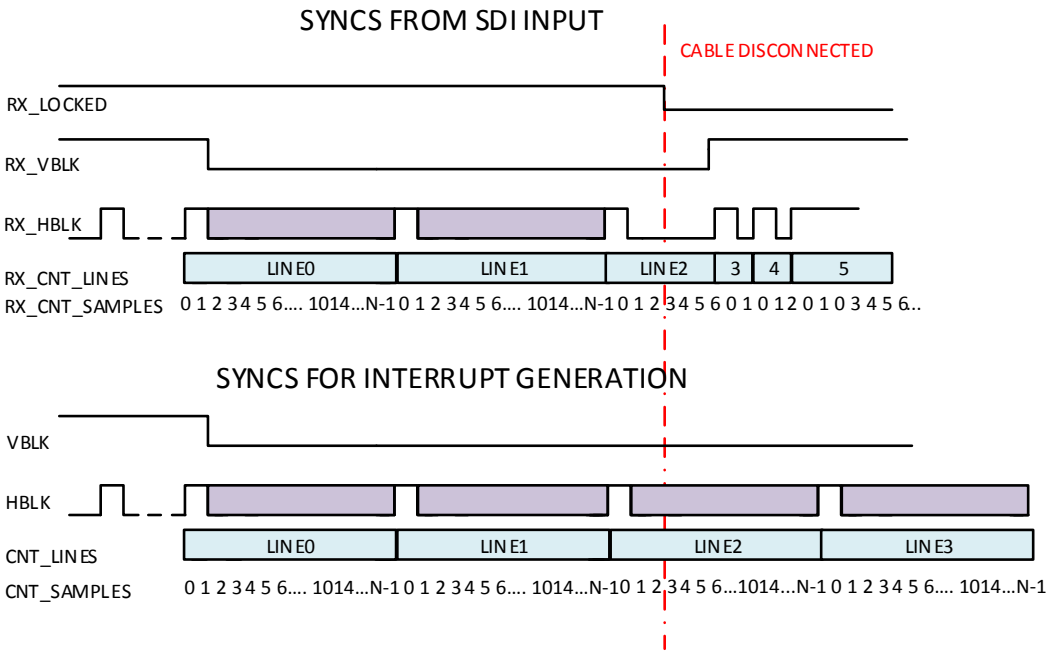


Figura 6.21. Generación de sincronismos tras desconexión del cable de entrada.

El funcionamiento del diseño en caso de que el módulo de recepción SDI se desenganche de la señal de entrada o en el caso de recibir un formato no soportado, es equivalente al caso de desconexión del cable de entrada.

Cuando la señal de entrada vuelve a conectarse con un formato válido, la FPGA no conmuta automáticamente a la señal de entrada. Se continúan generando sincronismos conforme a la señal de vídeo de color negro, y se informa al driver de que una nueva señal de entrada ha sido conectada mediante una interrupción MSI. Además, se proporciona información sobre el formato de vídeo recibido. Si el driver considera válido el formato de entrada, deberá indicar a la FPGA que conmute a dicha señal.

Podría ocurrir el caso, de que la nueva señal sea de otro formato distinto al de la señal de color en negro. En este caso, el pipeline debería ser destruido y liberar todos los recursos utilizados. Posteriormente, se deberá crear un nuevo pipeline conforme al nuevo formato de vídeo. Como el pipeline se detiene y se crea uno nuevo, el cambio en la cadencia en las interrupciones debido a que el formato de entrada nuevo es distinto al de la señal de color negro que se estaba generando, no supone ningún problema.

El cambio de formatos de vídeo en la señal de entrada sin desconectar el cable de entrada, es tratado de la misma manera. Se informa desde la FPGA al driver de que hay un nuevo formato a la entrada, la FPGA conmuta automáticamente al reloj del PLL para proporcionar una señal de color negro conforme a los sincronismos del anterior formato de vídeo y no conmutará a la nueva señal de entrada (de otro formato de vídeo) hasta que el driver se lo indique.

6.3.5.3 Generación de TLP en la escritura de vídeo

La dirección base de los distintos búferes es proporcionada por el driver en el registro correspondiente. El búfer al que acceder es indicado por el módulo BUFFER_MANAGER (Figura 6.17), por lo que cuando comienza la transferencia de un nuevo plano de vídeo se registra la dirección indicada para el búfer que proceda. A partir de esta dirección base se generarán los TLP necesarios hasta completar el plano de vídeo. Cada vez que una línea de vídeo es almacenada en la memoria interna de la FPGA, se indica que está lista para su transmisión a través del bus PCIe. Como la longitud de una línea de vídeo es superior al máximo tamaño de payload soportado (256 bytes), la línea es dividida en varios TLP de tamaño 256 bytes. Por lo tanto, la dirección entre los distintos TLP que componen una línea se incrementa en 256 bytes, y entre distintas líneas, en la longitud de una línea. Como se ha comentado previamente, en el caso de formatos de vídeo SD el salto entre líneas es de 768 bytes, para que los TLP estén alineados a 256 bytes y no se crucen 4 KB-boundaries.
En la Figura 6.22 se muestra el código perteneciente al primer dato de 128 bits que se proporciona al core PCIe para que genere el inicio de un nuevo TLP. El core PCIe exige paquetes de 128 bits para alcanzar la máxima tasa de transferencia. El resto de datos de 128 bits del TLP hasta completar los 256 bytes son el vídeo que se desea almacenar en memoria. El último paquete de 128 bits de una línea podría contener DW no válidos, este hecho se indica al core PCIe con una señal de habilitación a nivel de DW y rellenando los DW vacíos con el valor 0xD0DAD0DA.

```
trn_td          <= { {1'b0},
                     {`BMD_128_MWR_FMT_TYPE},
                     {1'b0}, //Reserved
                     1'b0, //TC
                     {4'b0}, //Reserved
                     1'b0,   //TD: There is no extra CRC on the TLP data
                     1'b0, //EP:0
                     {mwr_relaxed_order_i, mwr_nosnoop_i}, // 2'b00,
                     {2'b0}, //Reserved
                     {1'b0, tlp_size_word},//Number of TLP DWs
                     {completer_id_i[15:3], mwr_func_num}, //Identifies the sender
                     //TAG is ignored for TRD writes
                     {`TAG_VIDEO_0, cur_wr_count[4:0]},
                     (tlp_size_word == 1'b1) ? 4'b0 : mwr_lbe_i,
                     mwr_fbe_i,
                     // CN - Logic for lower QW
                     tmwr_addr[31:2],
                     {2'b0},
                     mwr_videodata0
                     };
```

Figura 6.22. Código de generación del primer paquete de 128 bits de un TLP de escritura BMD.

Los campos que componen el paquete de 128 bits que aparece en la Figura 6.22 se definen de la siguiente manera:

- *BMD_128_MWR_FMT_TYPE* indica que se trata de una escritura en memoria. En una transferencia de tipo *no_snoop (mwr_nosnoop_i = 1)*, la petición de escritura puede ser directamente atendida por el controlador DRAM para almacenar el dato, de forma que la caché del procesador no necesita ser invalidada. De esta forma, se reduce la latencia en los accesos. Esta opción es configurable mediante un registro de control, por lo que se deja que el driver decida cómo manejar las operaciones sobre caché.
- *mwr_relaxed_order_i* toma el valor 0, para indicar que los paquetes son transmitidos en orden consecutivo. Éste es el modo por defecto de ordenamiento conocido como *PCI Strongly Ordered Model*. Existen modos de ordenamiento que no aseguran que se envíen consecutivamente los paquetes, con vistas a incrementar la tasa de transferencia de datos, pero no han sido implementados.
- *tlp_size_word* es el tamaño del TLP en DWs.
- La concatenación de *completer_id_i* y *mwr_func_num*, identifican inequívocamente a la FPGA como emisor.
- El valor del TAG es ignorado en escrituras, pero se conforma por posibles tareas de depuración en el microprocesador con un identificador asociado a los paquetes de vídeo (*TAG_VÍDEO*) concatenado con un contador de los TLP enviados.
- *mwr_lbe_i*, indica qué bytes dentro del primer DW son válidos para ser escritos, en el diseño realizado siempre son cuatro (4' hF).
- *mwr_fbe_i*, indica qué bytes son válidos dentro del último DW.
- *tmwr_addr* es la dirección de escritura del TLP.
- *mwr_vídeodata0* son los primeros 32 bits de vídeo válido para ser almacenados en memoria.

6.3.6 Audio

La descripción de la aplicación desarrollada se ha centrado en la parte referente al procesamiento de vídeo, pero debido a que un sistema dedicado exclusivamente a vídeo carece de una utilidad práctica, se decidió incluir el procesamiento de hasta 4 canales de audio (0-4), extraídos de la señal de entrada SDI. La descripción de la demultiplexación y el código necesario para extraer el audio se puede encontrar en [10]. La parte de almacenamiento de audio y transmisión al microprocesador a través del bus PCIe está basada en el diseño realizado para vídeo. Como peculiaridades a resaltar, cada pareja de audios (canales de audio 0-1 y canales 2-3) son tratados como canales DMA independientes. Por lo tanto, habrá un canal DMA para la pareja 0-1 y otro canal DMA, para la pareja 2-3.

El tamaño de cada búfer de audio es de 384 muestras de audio de 32 bits (aunque son válidos únicamente 24 bits), incluyendo en cada búfer muestras intercaladas referentes a dos canales de audio (0 y 1, o 2 y 3). Únicamente se ha proporcionado soporte a la frecuencia de muestreo de audio de 48 KHz, por lo que la interrupción de finalización de escritura en búfer de audio se produce cada 4 ms.

Por otro lado, la frecuencia del reloj de audio que se utiliza es de 24.576 MHz derivado del reloj RX Clock. Cuando se produce una desconexión del cable, la entrada SDI no se encuentra enganchada o se está recibiendo un formato de vídeo o audio no soportado, se generan muestras de audio de silencio siguiendo la cadencia de los 4 ms indicados. Para poder seguir generando las interrupciones cuando el cable se desconecta y el reloj RX Clock deja de generarse, se requiere el reloj de 24.576 MHz que proviene del PLL externo que genera los relojes de vídeo.

6.4 Resultados

En este apartado se muestran los resultados obtenidos referentes a la implementación realizada en la FPGA, en términos de recursos empleados, consumo de potencia y frecuencia de funcionamiento. Los resultados mostrados se corresponden con la aplicación desarrollada para la extracción del vídeo de la señal de entrada SDI y su transmisión a través del PCIe, y además incluye la demultiplexación de cuatro canales de audio extraídos de la señal de entrada SDI y su transmisión a través del bus PCIe.

En la Figura 6.23, se muestra un resumen de los recursos utilizados. Como se puede apreciar en los gráficos que aparecen en la Figura 6.23, se han utilizado dos de los cuatro transceptores de señal (GT) disponibles en la Artix-7 15T, el 80% de la memoria interna disponible (*Block RAM, BRAM*), el 59.30 % de los flip-flops disponibles (*FF*), el 15.56% de los multiplicadores existentes (DSP) y 58 de los 150 pines disponibles (IO). El número de unidades lógicas utilizadas (*Look-up Table, LUT*) es del 97.4 %, por lo que si se desea extender la funcionalidad de la FPGA, el margen de ampliación es bastante pequeño. Para la generación de relojes internos en la FPGA se han utilizado dos módulos MMCM (*Mixed-Mode Clock Manager*) de los cinco disponibles. Adicionalmente, se ha utilizado el único módulo HW PCIe que existe y el módulo XADC, el cual permite la monitorización de la temperatura y de las tensiones internas en la FPGA.

Resource	Utilization	Available	Utilization %
LUT	10130	10400	97.40
LUTRAM	209	9600	2.18
FF	12334	20800	59.30
BRAM	20	25	80.00
DSP	7	45	15.56
IO	58	150	38.67
GT	2	4	50.00
MMCM	2	5	40.00

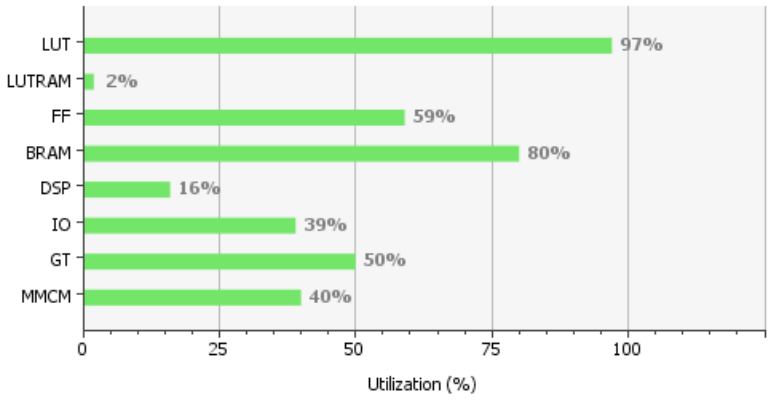


Figura 6.23. Recursos empleados en la FPGA Artix-7 15T.

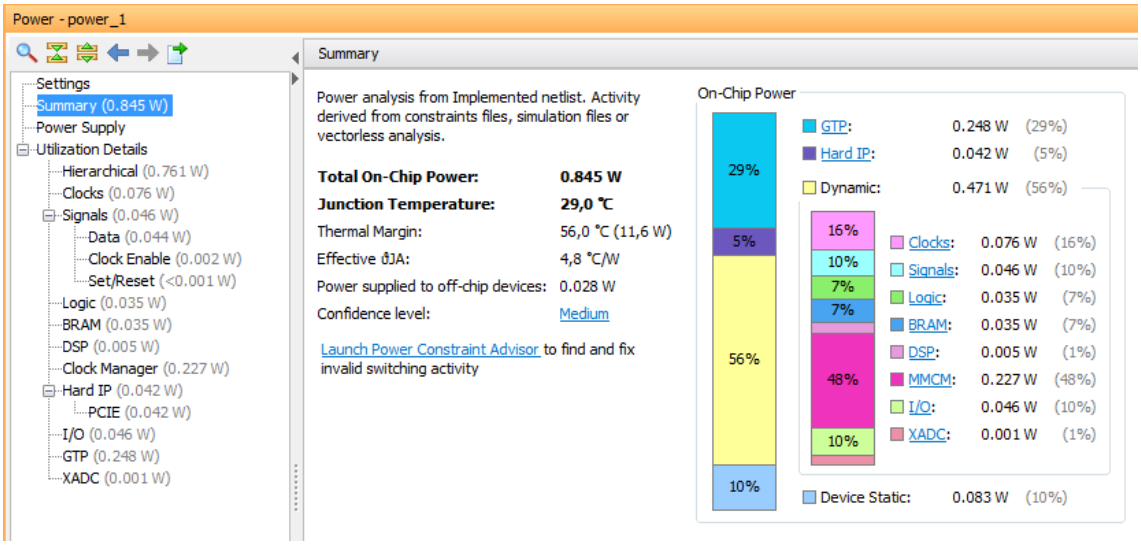


Figura 6.24. Reporte de consumo de potencia asociado al diseño realizado en la FPGA.

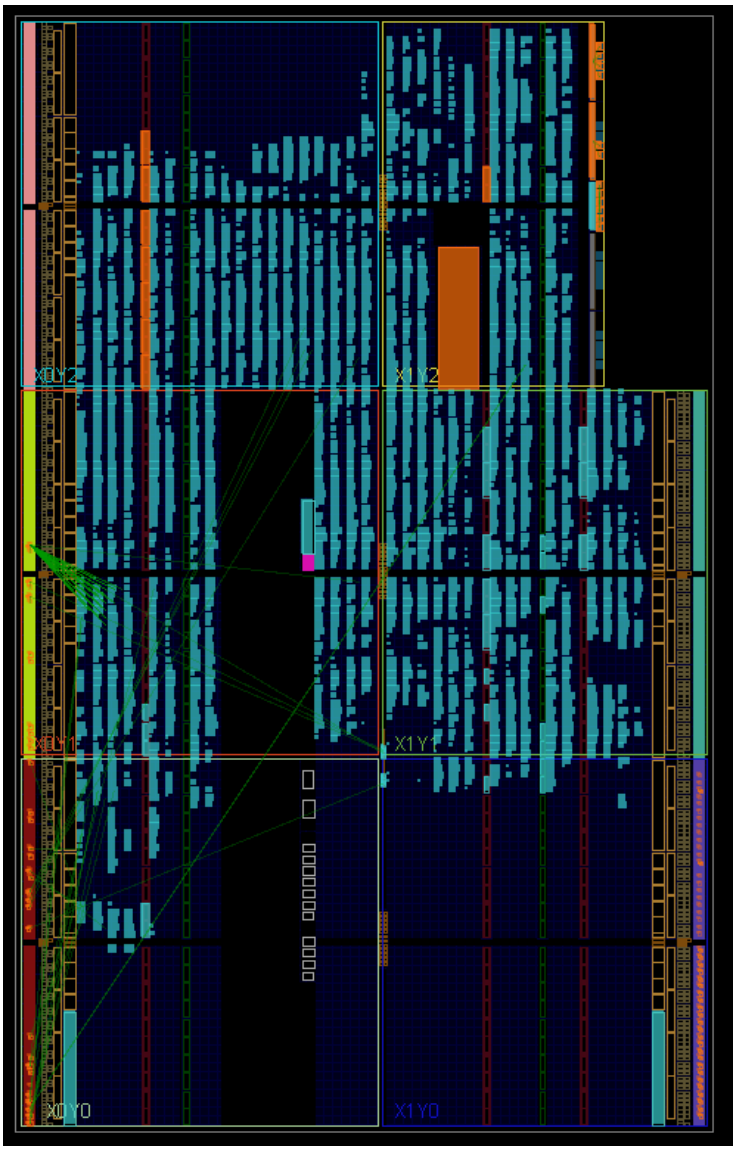


Figura 6.25. Layout de la implementación realizada.

Como se puede apreciar en la Figura 6.24, el consumo total de la FPGA asciende a escasos 0.845 W. El 29% de esta cantidad (0.248 W) se debe a los GTP encargados de la recepción SDI y a la comunicación PCIe. Un 10% del consumo (0.083W) está asociado al consumo estático de la FPGA, es decir el consumo que tiene la FPGA sin estar realizando ninguna operación. El 5% (0.042 W) está asociado al módulo hardware PCIe (*Hard IP*). Finalmente, el resto del consumo (*Dynamic*) se corresponde a un 56% (0.471 W) que se distribuyen entre la lógica utilizada para implementar el diseño, los relojes utilizados, la memoria interna empleada, dispositivos MMCM, DSP y a la conmutación de los pines de entrada/salida (I/O). El módulo XADC presenta un consumo prácticamente despreciable (0.001 W).

El *layout* de la FPGA una vez finalizada la implementación del diseño efectuado en la FPGA puede observarse en la Figura 6.25.

El diseño cumple holgadamente con las restricciones de frecuencias máximas de funcionamiento impuestas y las interacciones entre los distintos dominios de reloj han sido cuidadosamente tratadas en el fichero de restricciones correspondiente. En la Figura 6.26

se muestra un diagrama que muestra las diferentes redes de reloj existentes y las interacciones entre ellas.

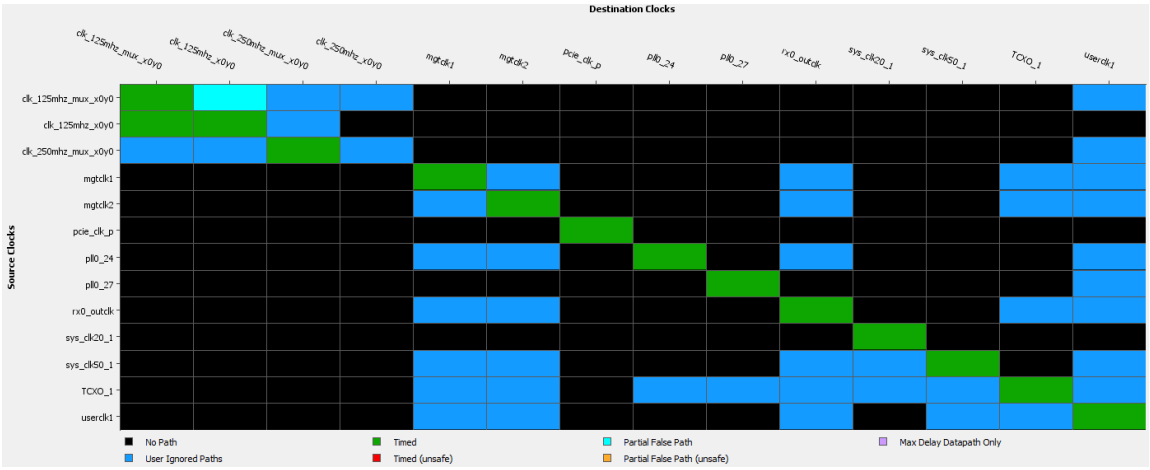


Figura 6.26. Redes de reloj presentes en el diseño e interacciones entre ellas.

La máxima frecuencia de reloj utilizada en el diseño es de 250 MHz, reloj necesario para el diseño implementado que posibilita la comunicación PCIe. Los módulos asociados a la comunicación PCIe también requieren un reloj de usuario de 125 MHz. Los módulos asociados a la recepción SDI deben de ser capaces de soportar una frecuencia máxima de funcionamiento de 148.5 MHz, mientras que la frecuencia de reloj de los módulos asociados al audio es de 24.576 MHz. Finalmente, se utilizan dos relojes destinados a labores de programación de los dispositivos externos a la FPGA mediante buses *SPI* (*Serial Peripheral Interface*) e I^2C (*Inter-Integrated Circuit*) de 20 y 50 MHz respectivamente.

6.5 Referencias

[1] PCI-SIG. (March 4, 2009). PCI Express® Base Specification". Revision 2.1.

[2] Mellanox Technologies. (Febrero de 2019). Understanding PCIe Configuration for Maximum Performance. Recuperado de https://community.mellanox.com/s/article/understanding-pcie-configuration-for-maximum-performance

[3] Xilly. (Enero de 2019). Down to the TLP: How PCI express devices talk (Part I). Recuperado de http://xillybus.com/tutorials/pci-express-tlp-pcie-primer-tutorial-guide-1

[4] Mohammadi N. (March 2015). Texas Instruments. SDI Video Bit Rate Calculation. Application Report SNLA232. Recuperado de http://www.ti.com/lit/an/snla232/snla232.pdf

[5] Xilinx. (Enero de 2019). All Programmable 7 Series Product Selection Guide. Recuperado de https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf

[6] Xilinx. (December 19, 2016). 7 Series FPGAs GTP Transceivers User Guide. UG482 (v1.9).

[7] Xilinx. (November 10, 2015). Implementing SMPTE SDI Interfaces with Artix-7 FPGA GTP Transceivers. XAPP1097 (v1.0.1).

[8] Xilinx. (Enero de 2019). Recuperado de https://secure.xilinx.com/webreg/clickthrough.do?cid=344558

[9] Texas Instruments. (August 2015). LMH0387 3 Gbps HD/SD SDI Configurable I/O Adaptive Cable Equalizer / Cable Driver. SNLS315H.

[10] Xilinx. (November 9, 2009). Audio/Video Connectivity Solutions for Virtex-5 FPGAs Reference Designs for the Broadcast Industry 2. XAPP1014 (v1.2).

[11] International Telecommunication Union. (January 2012). Payload identification data structure for digital television interfaces. Recommendation ITU-R BT.1614.

[12] Xilinx. (January, 2010). Dynamically Programmable DRU for High-Speed Serial I/O. XAPP875 (v1.1).

[13] Xilinx. (December 5, 2018). 7 Series FPGAs Integrated Block for PCI Express v3.3 LogiCORE IP Product Guide. Vivado Design Suite. PG054.

[14] Lawley J. (April 3, 2015). Xilinx. Application Note: 7 Series, Virtex-6, Virtex-5, Spartan-6 and Spartan-3 FPGAs. Bus Master Performance Demonstration Reference Design for the Xilinx Endpoint PCI Express Solutions. XAPP1052 (v3.3).

Capítulo 7

Conclusiones y futuras líneas de trabajo

7.1 Conclusiones

El trabajo realizado ha permitido la consecución con éxito del principal objetivo, ya que se han diseñado e implementado un conjunto de algoritmos que consiguen mejorar la eficiencia en el proceso de codificación HEVC, logrando una importante reducción de los tiempos destinados a la codificación con un leve incremento en la tasa de datos generada y una imperceptible pérdida de calidad visual. Los objetivos específicos definidos inicialmente también se han alcanzado:

1. Obteniendo un alto nivel de comprensión de todas las herramientas y técnicas que componen el estándar HEVC. Se ha realizado un breve resumen de algunas de las más importantes en el contenido de la memoria.
2. Se ha estudiado el estado del arte y se han realizado aportaciones significativas en el ámbito de la codificación HEVC, tal y como demuestran las publicaciones realizadas donde se incluye una comparativa con trabajos relevantes de la misma temática. En dichas comparativas el flujo de codificación propuesto alcanza una de las mejores relaciones de compromiso entre reducción de tiempo de codificación obtenido y valor BD-Rate asociado.
3. Se han diseñado, evaluado e implementado algoritmos de optimización con tal grado de abstracción que pueden ser utilizados en diversas plataformas hardware.

4. Establecimiento de la plataforma hardware más adecuada para la implementación de los algoritmos.
5. La integración de los algoritmos propuestos en un software propietario de una empresa líder en el sector de la producción de equipos profesionales de audio y vídeo para el mercado broadcast, ha posibilitado la creación de una nueva familia de productos basados en el estándar HEVC.

La memoria contiene una descripción de la evolución de los estándares de compresión de vídeo a lo largo de la historia, define la importancia de la estandarización y la necesidad de la utilización de la compresión de vídeo en la actualidad, ya que su evolución está directamente ligada con la expansión de nuevas tecnologías y dispositivos. Se han descrito las técnicas utilizadas por los diferentes estándares y cómo han evolucionado para incrementar la tasa de compresión. Se han resumido las mejoras que ha introducido HEVC respecto a su predecesor H.264, lo que ha permitido establecer una base teórica para luego abordar su optimización. Por lo tanto, la memoria realizada puede ser utilizada como un medio de iniciación en el campo de la compresión de vídeo y una introducción al estándar HEVC.

El estudio realizado sobre la literatura existente ha permitido descartar ciertas líneas de trabajo debido entre otras causas, a la carga computacional que suponen o a no posibilitar la aplicación de cómputo paralelo. Los algoritmos presentados se basan en una etapa de análisis de la imagen que permite extraer las características del vídeo de entrada a la codificación. En concreto, las aportaciones realizadas se pueden resumir en las siguientes:

1. Se realiza un análisis de la imagen de entrada mediante las unidades lógicas de la GPU para detectar regiones que presentan texturas homogéneas.
2. Se aplica una estimación de movimiento sobre la imagen de entrada para obtener un mapa de vectores de movimiento que permita detectar regiones temporalmente homogéneas. La estimación de movimiento se realiza invocando módulos hardware específicos de las GPU de Intel. El análisis del mapa de vectores de movimiento y posterior clasificación, se realiza empleando las unidades lógicas de la GPU.
3. Detección de regiones espacialmente homogéneas en las cuales existe una direccionalidad espacial predominante haciendo uso de un módulo de hardware dedicado presente en las GPU de Intel destinado a obtener los mejores modos de predicción intra y tamaños de bloque, bajo el estándar H.264. El análisis de los datos, la adaptación al estándar HEVC y la posterior clasificación, son realizados empleando las unidades lógicas de la GPU.
4. Optimización del flujo de codificación en regiones que presentan texturas homogéneas:
    a. Algoritmo de decisión de tamaño de bloque en CU de tipo intra e inter.
    b. Algoritmo de decisión de modo de predicción intra.
5. Optimización del flujo de codificación en CU tipo inter que presentan homogeneidad temporal:
    a. Algoritmo de decisión de tamaño de bloque y de decisión de modo de partición inter.
    b. Un algoritmo de configuración adaptativa de la ventana de búsqueda para la estimación de movimiento.
6. Optimización del flujo de codificación en CU tipo intra que presentan una direccionalidad espacial predominante:
    a. Algoritmo de decisión de tamaño de bloque.

   b. Algoritmo optimizado de decisión de modo de predicción intra utilizando información derivada de la predicción intra H.264.
7. Establecimiento de una parada prematura en el flujo de codificación que impide la ejecución de la predicción inter. Dicha parada está basada en la comparación de los costes obtenidos para la predicción intra e inter por los módulos hardware H.264.
8. Mejoras subjetivas de la calidad visual por medio de una reducción del valor de cuantificación en regiones con texturas homogéneas y en una etapa de pre-procesamiento de la imagen basada en la reducción de información redundante en regiones con estructura caótica.
9. Algoritmo de reducción de artefactos intrínsecos al HEVC.
10. Evaluación de la calidad y determinación de umbrales basándose en el funcionamiento del sistema visual humano.
11. Implementación en un sistema de codificación en tiempo real.

Trabajar con la imagen de entrada permite independizar los procesos de análisis y clasificación del proceso de codificación, de manera que puedan ser implementados en un dispositivo ajeno a la codificación trabajando como co-procesador. Para demostrar esta asunción en el sistema de codificación propuesto, estos procesos han sido implementados en una GPU, mientras que la codificación HEVC se realiza en un microprocesador. Los algoritmos asociados han sido implementados utilizando lenguaje OpenCL, lo que proporciona en el código una interoperabilidad que permite que sean ejecutados en otros dispositivos.

Los principales trabajos realizados en el ámbito de la compresión de vídeo se centran en desarrollar algoritmos implementados en el software de referencia HM que se distribuye junto a la norma ISO/IEC 23008-2 (HEVC). El software HM, debido al enorme tiempo que destina a la codificación, puede llegar a enmascarar que el tiempo de ejecución de los propios algoritmos es inabordable en implementaciones destinadas a ser ejecutadas en tiempo real. Existe un vacío dentro de la literatura existente sobre implementaciones que realmente puedan ser integradas en dispositivos de bajo coste, ejecutadas en tiempo real y que proporcionen las altas prestaciones que se requieren para el desarrollo de equipos profesionales de transmisión de vídeo. El uso de arquitecturas híbridas y cómputo heterogéneo permite manejar tareas computacionalmente muy intensas utilizando plataformas de bajo consumo y coste. Partiendo de esta base, se ha creado un sistema de codificación HEVC en tiempo real constituido por una FPGA, una GPU y un microprocesador. La FPGA realiza la captura de una señal SDI, extrae el vídeo y cuatro canales de audio, y los transmite a través del bus PCIe a un microprocesador. En la GPU embebida del microprocesador se ha implementado una conversión del tipo de muestreo de color para adaptar el vídeo de entrada al muestreo de color que requiere el codificador y los algoritmos de análisis de la imagen. La GPU incluye una etapa de análisis de la imagen y posterior clasificación de regiones temporal y espacialmente homogéneas. En la GPU también es posible realizar un cambio de la resolución de la imagen y habilitar un filtrado de la imagen para eliminar la información redundante. La imagen procesada y el conjunto de datos resultado del análisis y clasificación realizada, son compartidos con el procesador a cargo de realizar la codificación HEVC. En el procesador haciendo uso de los datos obtenidos en la GPU, se implementa un flujo de codificación HEVC optimizado. La integración de los algoritmos propuestos bajo las condiciones de test que se utilizan en la literatura relacionada con la codificación HEVC para establecer comparativas, demuestran la eficiencia y eficacia del flujo propuesto. Para facilitar la comparativa con otros trabajos se han establecido gráficos haciendo uso de Diagramas de Pareto, de esta forma los trabajos que pertenecen al frente de Pareto pueden ser considerados como aquellos que

proporcionan una relación de compromiso óptima entre el valor de BD-Rate y la reducción de tiempos de ejecución obtenidos. Como muestran dichos gráficos, las soluciones propuestas siempre se encuentran dentro del frente de Pareto, proporcionando valores muy bajos de BD-Rate frente a una importante reducción del tiempo de codificación.

A modo de resumen, las soluciones actuales basadas en HEVC pueden ser clasificadas en dos grupos:

1. Las que proporcionan gran calidad visual a expensas de un enorme gasto computacional, consumo de potencia y elevado coste.
2. Soluciones de bajo precio que sacrifican el grado de compresión de datos y la calidad visual del sistema para utilizar plataformas hardware de coste reducido.

Ambos tipos de soluciones pueden beneficiarse de los algoritmos desarrollados, puesto que aportan una considerable reducción de la carga computacional asociada al proceso de codificación a expensas de un leve incremento en la tasa de datos obtenida y de una pérdida en la calidad visual imperceptible.

7.2 Futuras líneas de trabajo

El código OpenCL utilizado para implementar los algoritmos de análisis y clasificación de la imagen proporciona una interoperabilidad que permite que sean ejecutados en diversos dispositivos. En los resultados mostrados se ha utilizado una GPU, siendo una futura línea de trabajo la implementación de estos procesos en la FPGA destinada a extraer el audio y el vídeo de la entrada SDI del sistema y destinar a la GPU a otras tareas.

La primera versión del estándar de compresión de vídeo VVC, considerado la evolución de HEVC, será liberado en 2020 y paulatinamente irá sustituyendo a HEVC. VVC mejora el proceso de división de la imagen recursivo que incorpora HEVC añadiendo tamaños de bloque de 128x128 píxeles y nuevos modos de partición inter. Aunque la incorporación de estos modos mejora las prestaciones en la codificación, se añade complejidad al proceso, por lo que se incrementan los tiempos de ejecución asociados. Por lo tanto, los algoritmos de decisión de tamaño de bloque como los presentados en este trabajo serán de gran utilidad para su optimización. Tanto la predicción intra como la inter están igualmente basadas en HEVC, por lo que las optimizaciones realizadas sobre estos procesos son totalmente válidas en VVC. Los algoritmos de mejora de calidad visual se basan en conceptos que también pueden reutilizarse en el nuevo estándar. Debido a la existencia de estas compatibilidades entre los dos estándares, una importante e innovadora línea futura de investigación es la integración de todos los algoritmos desarrollados en la presente Tesis en el nuevo estándar VVC.

La creciente expansión de los algoritmos basados en inteligencia artificial en todos los ámbitos de la tecnología, no debe de pasar desapercibida en la compresión de vídeo digital. La detección de las características presentes en la imagen es susceptible a optimización mediante algoritmos que aplicando técnicas de *machine learning* sean capaces de mejorar la clasificación, mediante una etapa inicial de aprendizaje supervisado que evolucione a que el algoritmo sea capaz de mejorar la detección de forma autónoma. Por otro lado, se ha comenzado a trabajar en algoritmos que sean capaces de establecer automáticamente la configuración óptima a aplicar en la codificación, en función de

parámetros básicos como el tamaño de bitstream generado, calidad visual proporcionada, y fundamentalmente la carga computacional, consumo y temperatura. La idea principal es proporcionar la mayor calidad visual y menor tasa de datos generada en función de la carga computacional del sistema. Cuando la carga es excesiva, se deja de codificar en tiempo real, por lo que el algoritmo de control debe de ser capaz de prever esa situación antes de que ocurra y actuar en consecuencia. Es posible reducir la carga computacional aumentando el número de optimizaciones que se aplican y reduciendo la complejidad de las técnicas empleadas de manera predictiva, cuando se observa que la temperatura y consumo del procesador se incrementan. De esta forma, se reduce la calidad visual proporcionada, pero esto es preferible a sobrecargar el sistema y provocar un funcionamiento erróneo.

Glosario

AMP	Asymmetric Motion Partitioning
AOMedia	Alliance for Open Media
API	Application Programming Interface
ArTeCS	Group of Architecture and Technology of Computing *Systems*
AVC	Advanced Video Coding
AVX	Advanced Vector extensions
AV1	AOMedia Video 1
BAR	Base Address Register
BD-Rate	Bjontegaard Delta Rate
BMD	Bus Master DMA
BRAM	Block RAM
CABAC	Context Adaptative Binary Arithmetic Coder
CAE	Context-Content Aware Encoding
CAE²	Context-based Arithmetic Coding
CAVLC	Context-Adaptative Variable Length Code
CB	Coding Block
CCIT	Comité Consultatif International Téléphonique et Télégraphique
CCTV	Circuito Cerrado de Televisión
CU	Coding Unit
CRC	Cyclic Redundancy Check
CTU	Coding Tree Unit
CFE	Call for Evidence
CFP	Call for Proposal
DMA	Direct Memory Access
DRP	Dynamic Reconfiguration Port
DW	Double Word
BO	Band Offset
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DSIS	Double Stimulus Impairment Scale
DST	Discrete Sine Transform
DWT	Discrete Wavelet Transform
EO	Edge Offset

FPGA	Field-Programmable Gate Array
FPS	Frames per second
GPU	Graphics Processing Unit
HDR	High Dynamic Range
HEVC	High Efficiency Video Coding
HPEL	Half-Pixel Interpolation
HVS	Human Visual System
HW	Hardware
IBF	Intra-Boundary Filter
IETF	Internet Engineering Task Force
IFC	Information Fidelity Criterion
ILP	Instruction Level Parallelism
I^2C	Inter-Integrated Circuit
ISO	International Standards Organization
ITU	International Telecommunication Union
JCT-VT	Joint Collaborative Team on Video Coding
JEM	Joint Video Exploration Model
JPEG	Joint Photographic Experts Group
JVET	Joint Video Exploration Team
JVT	Joint Video Team
LUT	Look-Up Table
MAE	Mean Absolute Error
MGT	Multi-Gigabit Transceiver
MMCM	Mixed-Mode Clock Manager
MOS	Mean Opinion Score
MPEG	Motion Picture Experts Group
MPM	Most Probable Modes
MSE	Mean Square Error
MSI	Message Signaled Interrupt
MS-SSIM	Multi Scale SSIM
MVD	Motion Vector Difference
MVP	Motion Vector Predictor
NETVC	Internet Video Codec
PB	Prediction Block
PCIe	Peripheral Component Interconnect Express
PCI-SIG	PCI Special Interest Group
PIO	Programmed Input/Output
POC	Picture Order Count
PPM	Parts Per Million
PSNR	Peak Signal-to-Noise
PU	Prediction Unit
QP	Quantization Parameter
QPEL	Quarter-Pixel Interpolation
RC	Rate Control
RDO	Rate-Distortion Optimization
RDPCM	Residual Differential Pulse Code Modulation
RExt	Range Extensions of the HEVC standard
RMD	Rough Mode Decision
RQT	Residual Quadtree
SAE	Sum of Absolute Errors

SAD	Sum of Absolute Differences
SAO	Sample Adaptive Offset
SATD	Sum of Absolute Transformed Differences
SDI	Serial Digital Interface
SD-TV	Standard Definition Television
SIMD	Single Instruction, Multiple Data
SMPTE	The Society of Motion Picture and Television Engineers
SPI	Serial Peripheral Interface
ST-SSIM	Spatial-Temporal SSIM
SSE	Streaming SIMD Extensions
SSIM	Structural Similarity Index Measure
SSIMb	Structural Similarity Block Index Measure
TAC	Tomografía Axial Computarizada
TB	Transform Block
TCIA	The Cancer Imaging Archive
TDT	Televisión Digital Terrestre
TS	Transform Skip
TU	Transform Unit
UHD-TV	Ultra High Definition Television
VCEG	ITU-T Video Coding Expert Group
VIF	Visual Information Fidelity
VIFP	Visual Information Fidelity Pixel
VLC	Variable-length coding
VLIW	Very Long Instruction Word
VQM	Video Quality Metric
VVC	Versatile Video Coding
VTM	VVC Test Model
WPP	Wavefront Parallel Processing
3-*SSIM*	Three-Component Weighted Structural Similarity

Apéndice 1

Vectores de Interrupción MSI

Evento	Vector de Interrupción
Entrada SDI0 (RX0). Desconexión/conexión de cable o cambio en la señal de LOCK.	0
Fin de escritura de un plano de vídeo en búfer del host (RX0)	1
Fin de escritura de una trama de audio de los canales 0/1 en búfer del host (RX0)	2
Fin de escritura de una trama de audio de los canales 2/3 en búfer del host (RX0)	3

FPGA Control Register 0: Current Version: Address: 0x000

Field	Bit(s)	Initial Value	R/RW
Version Number (char)	7-0	0x00	R
Version Number (first digit)	15-8	0x00	R
Version Number (second digit)	23-16	0x00	R
Version Number (third digit)	31-24	0x01	R

Version Number: Versión actual de FPGA. Ejemplo, “1.0.0.a”

FPGA Control Register 1 (DMA Control): Address: 0x004

Field	Bit(s)	Initial Value	R/RW
Write DMA No Snoop	0	0	RW
RX0 Locked	1	0	RW
RX0 - Write Vídeo DMA Interrupt Enable	2	0	RW
RX0 - Write Audio 0/1 DMA Interrupt Enable	3	0	RW
RX0 - Write Audio 2/3 DMA Interrupt Enable	4	0	RW

Reserved: Reservado para futuras aplicaciones.

Write DMA NoSnoop: '1' Habilita el atributo *No Snoop* en los Transaction Layer Packets (TLP). Una transferencia No Snoop es directamente almacenada en memoria sin intervenir la caché del procesador. En una transferencia de tipo No Snoop la petición de escritura puede ser directamente atendida por el controlador DRAM para almacenar el dato, de forma que la caché del procesador no necesita ser invalidada. De esta forma, se reduce la latencia en los accesos.

RX0-Locked: '1' Habilita la generación de interrupciones por parte de la FPGA cuando se detecta un cambio en la señal locked de la entrada SDI RX0. La señal Locked a '1' indica que se ha detectado un formato SDI válido a la entrada y que el módulo de recepción SDI está enganchado a la entrada.

RX0 - Write Vídeo DMA Interrupt Enable: '1' Habilita la generación de interrupciones por parte de la FPGA cuando finaliza la escritura de un búfer de vídeo (SDI RX0) en la memoria del host.

RX0 - Write Audio0/1 DMA Interrupt Enable: '1' Habilita la generación de interrupciones por parte de la FPGA cuando finaliza la escritura de un búfer de audio de los canales 0 y 1 (SDI RX0) en la memoria del host.

RX0 - Write Audio2/3 DMA Interrupt Enable: '1' Habilita la generación de interrupciones por parte de la FPGA cuando finaliza la escritura de un búfer de audio de los canales 2 y 3 (SDI RX0) en la memoria del host.

FPGA Control Register 2 (Device Link Width Status): Address: 0x008

Field	Bit(s)	Initial Value	R/RW
Capability Maximum Link Width	5: 0	1	R
Reserved	7:6	0	--
Negotiated Maximum Link Width	13:8	Negotiated Maximum Link Width	R
Reserved	31:14	0	R

Capability Maximum Link Width: Maximum Link Width (número de lanes) que soporta la FPGA.

1→x1; 2→x2; 4→x4; 8→x8

Negotiated Maximum Link Width: Link Width negociado entre microprocesador y FPGA.

1→x1; 2→x2; 4→x4; 8→x8

FPGA Control Register 3 (Device Link Transaction Size Status): Address: 0x00C

Field	Bit(s)	Initial Value	R/RW
Reserved	7:0	0	R
Programmed Max. Payload Size	10:8	Value	R
Reserved	15:11	0	R
Max. Read Request Size	18:16		
Reserved	31:19	0	R

Programmed Max. Payload Size: Máximo tamaño de payload que finalmente se utilizará al haberse negociado entre la FPGA y el microprocesador.

000b→128bytes; 001b→256bytes; 010b→512bytes; 011b→1024bytes;

100b→2048bytes; 101b→ 4096bytes; 110b,111b→Not supported

Maximum Read Request Size: El dispositivo no debe generar peticiones de lectura que excedan este tamaño.

000b→128bytes; 001b→256bytes; 010b→512bytes; 011b→1024bytes;

100b→2048bytes; 101b→ 4096bytes; 110b,111b→Not supported

FPGA Control Register 4 (System Status): Address: 0x010

Field	Bit(s)	Initial Value	R/RW
Device_temperature	11:0	0	R
VCC Internal	23:12	0	R

Device_temperature: Valor de temperatura de la FPGA. Realmente, se proporciona el valor que se obtiene del módulo XADC. La temperatura se deriva mediante la siguiente fórmula:

$$Temp\ (ºC) = = \frac{\text{XADC Code X } 503.975}{4096} - 273.15$$

VCC Internal: El XADC genera el código 0xFFF para 3V de voltaje. Por lo que la fórmula para obtener el valor de tensión correspondiente es:

(Voltage / 3) * 4096 = Code

(Code * 3) / 4096 = Voltage

Para el correcto funcionamiento de la FPGA el valor de VCCInternal debe de ser aproximadamente 1V.

FPGA Control Register 5 (System Status II): Address: 0x14

Field	Bit(s)	Initial Value	R/RW
VCC Aux	11:0	0	R
VCC BRAM	23:12	0	R

El XADC genera el código 0xFFF para 3V de voltage. Por lo que la fórmula para obtener el valor de tensión correspondiente es:

(Voltage / 3) * 4096 = Code

(Code * 3) / 4096 = Voltage

Para el correcto funcionamiento de la FPGA el valor de VCCAux debe de ser aproximadamente 1.8V.

Para el correcto funcionamiento de la FPGA el valor de VCCBRAM debe de ser aproximadamente 1V.

FPGA Control Register 6 (RX0 Status). Address: 0x018

Field	Bit(s)	Initial Value	R/RW
Reserved	0-4	-	-
RX0_change_done	5	0	R
RX0_change_fail	6	0	R
RX0_change_fail_code	9:7	0	R
RX0_CRC_error	10	0	R
RX0_error_cnt	26-11	0	R

RX0_change_done → '1' El cambio de formato en la entrada RX0 se realizó con éxito.

RX0_change_fail → '1' Se produjo un error en el cambio de formato a la entrada RX0, indicado por el campo RX0_change_fail_code

RX0_change_fail_code → Código del error producido.

RX0_CRC_error → Se produjo un error de CRC en la entrada RX0.

RX0_error_cnt → Contador de errores producidos en la entrada RX0.

FPGA Control Register 7 (MGT Resets). Address: 0x01C

Field	Bit(s)	Initial Value	R/RW
MGTRX0_clear_errors	0	0	RW
MGTRX0_reset	1	0	RW

MGTRX0_clear_errors → '1' Provoca reset de los flags y contadores de error asociados a la entrada RX0.

MGTRX0_reset →'1' Fuerza reset del MGT asociado a la recepción SDI RX0.

FPGA Control Register 8 (SDI_RX_Disconnected). Address: 0x020

Field	Bit(s)	Initial Value	R/RW
Cable Connected RX0	0	1	R

Cable Connected RX0→ Indica si en la entrada RX0 hay señal conectada ('1') o no ('0').

FPGA Control Register 9 (SDI PLLs status). Address: 0x024

Field	Bit(s)	Initial Value	R/RW
QPLL_locked	0	0	R
CPLL0_locked	1	0	R
CPLL1_locked	2	0	R

Señales de Locked de los diferentes PLLs de los MGT de la FPGA.

FPGA DMA0 Control Register 0 (SDI Rx0 Vídeo Enable). Address: 0x100

Field	Bit(s)	Initial Value	R/RW
Reserved	10:0	0	R
Enable PCIe Transmission	11	0	RW

Enable PCIe Transmission: '1' Habilita la transmisión de vídeo desde la FPGA a la memoria del microprocesador a través del PCIe por el canal RX0.

FPGA DMA0 Control Register 1 (SDI Rx0 Vídeo Overflow). Address: 0x104.

Field	Bit(s)	Initial Value	R/RW
Reserved	5:0	0	R
Buffer0 Overflow	6	0	R
Buffer1 Overflow	7	0	R
Buffer2 Overflow	8	0	R

BufferX Overflow: '1' La FPGA ha intentado escribir el búfer X sin que el host lo haya leído. Realmente, se produce este error si la FPGA tras recorrer todos los búferes con el flag *Enable Write* activo, comienza a escribir en el búfer X estando en este búfer el flag *Buffer Processed* activo.

FPGA DMA0 Control Register 2 (RX0 Vídeo - Write DMA Address0). Address: 0x108

Field	Bit(s)	Initial Value	R/RW
Enable Write	0	0	RW
Buffer Processed	1	0	RW
Write DMA Address0	31:2	0	RW

Enable Write: '1' Habilita la escritura en la posición de memoria Write DMA Address 0.

Buffer Processed: '1' El búffer fue correctamente escrito por la FPGA. Cuando el microprocesador atienda a este búfer debería escribir un '0' en este bit. Si no lo hace y la FPGA lo sobreescribe se generará un error de *Buffer Overflow*.

Write DMA Address0: Dirección del microprocesador en la que realizar la escritura (búfer 0).

FPGA DMA0 Control Register 3 (RX0 Vídeo- Write DMA Address1). Address: 0x10C

Field	Bit(s)	Initial Value	R/RW
Enable Write	0	0	RW
Buffer Processed	1	0	RW
Write DMA Address1	31:2	0	RW

Enable Write: '1' Habilita la escritura en la posición de memoria Write DMA Address 1.

Buffer Processed: '1' El búffer fue correctamente escrito por la FPGA. Cuando el microprocesador atienda a este búfer debería escribir un '0' en este bit. Si no lo hace y la FPGA lo sobreescribe se generará un error de *Buffer Overflow*.

Write DMA Address0: Dirección del microprocesador en la que realizar la escritura (búfer 1).

FPGA DMA0 Control Register 4 (RX0 Vídeo- Write DMA Address2). Address: 0x110

Field	Bit(s)	Initial Value	R/RW
Enable Write	0	0	RW
Buffer Processed	1	0	RW
Write DMA Address2	31:2	0	RW

Enable Write: '1' Habilita la escritura en la posición de memoria Write DMA Address 2.

Buffer Processed: '1' El búffer fue correctamente escrito por la FPGA. Cuando el microprocesador atienda a este búfer debería escribir un '0' en este bit. Si no lo hace y la FPGA lo sobreescribe se generará un error de *Buffer Overflow*.

Write DMA Address0: Dirección del microprocesador en la que realizar la escritura (búfer 2).

FPGA DMA0 Control Register 7 (SDI RX0 Timestamp - Vídeo Buffer 0 (LSB)). Address: 0x11C

Field	Bit(s)	Initial Value	R/RW
TimeStamp	31-0	0	RW

TimeStamp: Parte baja del timeStamp (unsigned int 64bits) asociado al búfer 0 definido para la recepción de vídeo SDI (RX0) en el registro de control FPGA DMA0 Control Register 2.

FPGA DMA0 Control Register 8 (SDI RX0 Timestamp - Vídeo Buffer 0 (MSB)): Address: 0x120

Field	Bit(s)	Initial Value	R/RW
TimeStamp	31-0	0	RW

TimeStamp: Parte alta del timeStamp (unsigned int 64bits) asociado al búfer 0 definido para la recepción de vídeo SDI (RX0) en el registro de control FPGA DMA0 Control Register 2.

FPGA DMA0 Control Register 9 (SDI RX0 Timestamp - Vídeo Buffer 1 (LSB)). Address: 0x124

Field	Bit(s)	Initial Value	R/RW
TimeStamp	31-0	0	RW

TimeStamp: Parte baja del timeStamp (unsigned int 64bits) asociado al búfer 1 definido para la recepción de vídeo SDI (RX0) en el registro de control FPGA DMA0 Control Register 3.

FPGA DMA0 Control Register 10 (SDI RX0 Timestamp - Vídeo Buffer 1 (MSB)): Address: 0x128

Field	Bit(s)	Initial Value	R/RW
TimeStamp	31-0	0	RW

TimeStamp: Parte alta del timeStamp (unsigned int 64bits) asociado al búfer 1 definido para la recepción de vídeo SDI (RX0) en el registro de control FPGA DMA0 Control Register 3.

FPGA DMA0 Control Register 11 (SDI RX0 Timestamp - Vídeo Buffer 2 (LSB)): Address: 0x12C

Field	Bit(s)	Initial Value	R/RW
TimeStamp	31-0	0	RW

TimeStamp: Parte baja del timeStamp (unsigned int 64bits) asociado al búfer 2 definido para la recepción de vídeo SDI (RX0) en el registro de control FPGA DMA0 Control Register 4.

FPGA DMA0 Control Register 12 (SDI RX0 Timestamp - Vídeo Buffer 2 (MSB)): Address: 0x130

Field	Bit(s)	Initial Value	R/RW
TimeStamp	31-0	0	RW

TimeStamp: Parte alta del timeStamp (unsigned int 64bits) asociado al búfer 2 definido para la recepción de vídeo SDI (RX0) en el registro de control FPGA DMA0 Control Register 4.

FPGA DMA0 Control Register 13-15

Reservados.

FPGA DMA0 Control Register 16 (SDI RX0 Vídeo Information). Address: 0x140.

Field	Bit(s)	Initial Value	R/RW
Rx Locked	0	0	R
RX Width	11:1	0	R
RX Height	22:12	0	R
Rx Rate	26:23	0	R

Rx Scan	27	0	R
Rx Level B	28	0	R
Rx Unsupported Vídeo Format	29	0	R

RX Vídeo Locked: ‘1’ Indica el módulo SDI se ha enganchado al vídeo de entrada. El valor del resto de campos de este registro (a excepción de *Rx Unsupported Format* y *Rx Link Status*) es únicamente válido cuando esta señal toma el valor ‘1’.

RX Width: Ancho de la imagen de entrada. (720,1280 o 1920)

RX Height: Alto de la imagen de entrada (576, 480, 720 o 1080)

RX Rate: Frame Rate

Value	**Frame Rate**	**Value**	**Frame Rate**
0000	None	0111	30
0010	23.98	1000	48
0011	24	1001	50
0100	49.95	1010	59.94
0101	25	1011	60
0110	29.97	Others	Reserved

RX Scan: ‘0’ Entrelazado, ‘1’ Progresivo

Rx Level B: ‘1’ si la entrada es 3G Level B.

Rx Unsupported Vídeo Format: ‘1’ si la entrada no es soportada.

Podría darse el caso (3G Level B): Rx Locked = ‘1’ y Rx Unsupported Format = ‘1’. Esta combinación de valores indicaría que el formato de entrada ha sido reconocido pero que no es soportado. Si Rx Locked = ‘0’ y Rx Unsupported Format = ‘1’, el formato de entrada se desconoce y por lo tanto, no se soporta.

FPGA DMA0 Control Register 17 (Black Signal Information): Address: 0x144

Field	Bit(s)	Initial Value	R/RW
Black Signal Locked	0	0	R
Black Signal Width	11:1	0	R
Black Signal Height	22:12	0	R
Black Signal Rate	26:23	0	R
Black Signal Scan	27	0	R

Black Signal Vídeo Locked: '1' Indica si se están proporcionando planos negros desde la FPGA al Host. El valor del resto de campos de este registro es únicamente válido cuando esta señal toma el valor '1'. La FPGA proporciona planos negros cuando el vídeo de entrada se ha desconectado o se ha detectado cambio de formato. El formato que proporciona es el formato del vídeo de entrada la última vez que se consideró válido (antes de desconectar o cambiar de formato).

Black Signal Width: Ancho de la imagen (720, 1280 o 1920)

Black Signal Height: Alto de la imagen (576, 480, 720 o 1080)

Black Signal Rate: Frame Rate

Value	Frame Rate	Value	Frame Rate
0000	None	0111	30
0010	23.98	1000	48
0011	24	1001	50
0100	49.95	1010	59.94
0101	25	1011	60
0110	29.97	Others	Reserved

RX Scan: ‘0’ Entrelazado, ‘1’ Progresivo**.**

FPGA DMA0 Control Register 18 (RX/Black Control): Address: 0x148

Field	Bit(s)	Initial Value	R/RW
Force RX	0	0	RW

Force RX: Si se asigna el valor ‘1’, la FPGA deja de proprocionar planos de vídeo negro y comienza a enviar el vídeo que se recibe por la entrada SDI (si éste es válido). La FPGA fuerza este flag a ‘0’ cuando ha atendido el comando.